Learning an Interpretable Traffic Signal Control Policy

James Ault
Texas A&M University
College Station, Texas
jault@tamu.edu

Josiah P. Hanna
University of Edinburgh
Edinburgh, U.K.
josiah.hanna@ed.ac.uk

Guni Sharon
Texas A&M University
College Station, Texas
guni@tamu.edu

ABSTRACT

Signalized intersections are managed by controllers that assign right of way (green, yellow, and red lights) to non-conflicting directions. Optimizing the actuation policy of such controllers is expected to alleviate traffic congestion and its adverse impact. Given such a safety-critical domain, the affiliated actuation policy is required to be interpretable in a way that can be understood and regulated by a human. This paper presents and analyzes several on-line optimization techniques for tuning interpretable control functions. Although these techniques are defined in a general way, this paper assumes a specific class of interpretable control functions (polynomial functions) for analysis purposes. We show that such an interpretable policy function can be as effective as a deep neural network for approximating an optimized signal actuation policy. We present empirical evidence that supports the use of value-based reinforcement learning for on-line training of the control function. Specifically, we present and study three variants of the *Deep Q-learning* algorithm that allow the training of an interpretable policy function. Our *Deep Regulatable Hardmax Q-learning* variant is shown to be particularly effective in optimizing our interpretable actuation policy, resulting in up to 19.4% reduced vehicles delay compared to commonly deployed actuated signal controllers.

KEYWORDS

deep reinforcement learning; interpretable; intelligent transportation

1 INTRODUCTION

Travel time studies in urban areas show that 12–55% of commute travel time is due to delays induced by signalized intersections (stopped or approach delay) [18, 35]. Hence, optimized signal controllers have the potential of reducing commute time, traffic congestion, emissions, and fuel consumption, while requiring minimal infrastructure changes.

Recent publications [17, 19, 36] proposed to utilize state–of–the–art reinforcement learning algorithms for online optimization of signal controllers. Such previous work showed a potential reduction of up to 73% in vehicle delays when compared to fixed–time actuation [25]. Despite showing compelling empirical results, the

controllers defined in such previous work have little applicability in the real world since their underlying control function is based on a deep neural network (DNN). While providing flexible and powerful function approximations, DNNs lack an interpretable inference process [29] which might prevent the implementation of related controllers in practice. Given their liability for drivers' safety and mobility, governmental transportation agencies are conservative in requiring that signal controllers are interpretable and regulated. Consequently, this paper focuses on defining and studying self–optimizing, regulatable signal control policies.

The following contributions are made, which, to the best of our knowledge, were not addressed in previous work:

(1) Define and justify a regulatable control function for the signal control domain.
(2) Define and study the effectiveness of a domain specific regulatable function when compared to a DNN–based policy.
(3) Study the effectiveness of different optimization methods for online training of a signal control policy. Namely, *Covariance Matrix Adaptation Evolution Strategy* (CMA–ES) [11], *Proximal Policy Optimization* (PPO) [31], and *Deep Q–learning* (DQN) [23].
(4) Develop three variants of the DQN algorithm that utilize and train a regulatable control function. These variants are denoted *Deep Regulatable Q–learning* (DRQ), *deep regulatable softmax Q-learning* (DRSQ), and *deep regulatable hardmax Q–learning* (DRHQ).
(5) Test the performance of the aforementioned optimization methods through computer–based simulation of a real–life intersection and observed demand.
(6) Compare the proposed methods with the commonly deployed *actuated signal controller* [7]. As opposed to previous work that were compared to, the less effective, fixed–timing actuation policy.

We show that a designed regulatable control function can reduce traffic delays by up to 30% when compared to common actuated signal controllers. Moreover, such a regulatable control function is shown to be competitive with the performance of a DNN based controller. Next, we turn to study online optimization approaches for signal controllers. Our empirical study shows that a value–based approach (specifically DQN) converges faster, and to a better policy, compared to a policy–gradient approach (specifically PPO). Our regulatable Q–learning variant, DRHQ, is shown to result in a policy that is competitive with commonly deployed actuated signal controllers after a single training episode. The control policy is further improved in successive episodes, reaching up to 19.4% reduced delays.
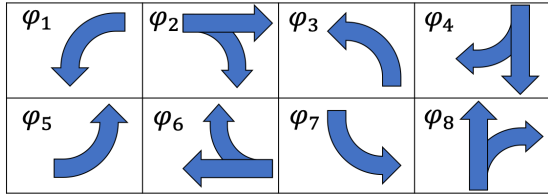
Figure 1: Common phase allocation for a 4–way intersection.

2 BACKGROUND AND RELATED WORK

This section provides the necessary background which includes the domain description, Markov decision processes, reinforcement learning, and related work.

2.1 The traffic signal domain

A signalized intersection is composed of incoming and outgoing roads where each road is affiliated with one or more lanes. Each signalized intersection is assigned a set of phases, Φ. Each phase, $\varphi \in \Phi$, is affiliated with a specific traffic movement through the intersection, as illustrated in Figure 1. Two phases are defined to be in *conflict* if they cannot be enabled simultaneously (their affiliated traffic movement is intersecting). For example, in the phase allocation presented in Figure 1, φ_2 and φ_1 are conflicting phases.

2.2 Reinforcement learning

In reinforcement learning (RL) an agent is assumed to learn through interactions with the environment. The environment is commonly modeled as a Markov decision process (MDP) which is defined by: $\mathcal{S}$ – the state space, $\mathcal{A}$ - the action space, $\mathcal{P}(s_t, a, s_{t+1})$ – the transition function of the form $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow Prob$, $R(s, a)$ – the reward function of the form $\mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, and γ – the discount factor. The agent is assumed to follow an internal policy π which maps states to actions, i.e., $\mathcal{S} \rightarrow \mathcal{A}$. The agent's chosen action (a_t) at the current state (s_t) affects the environment such that a new state emerges (s_{t+1}) as well as some reward (r_t) representing the immediate utility gained from performing action a at state s, given by $R(s, a)$. The observed reward is used to tune the policy such that the expected sum of discounted reward, $J_\pi = \sum_t \gamma^t r_t$, is maximized. The policy $argmax_\pi [J_\pi]$ is the optimal policy denoted π^*.

Recent publications [32, 36] suggest applying one RL approach, Deep Q–learning (DQN) [23], for training and operating signal controllers. In DQN, a deep neural network (DNN) is trained to map state–action pairs to a real number denoted the Q–value i.e., $\mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. The Q–value for a given state–action pair, (s, a), represents the sum of discounted reward after performing a at s and then following the optimal policy (π^*). Algorithm 1 presents the pseudocode for DQN (for now, ignore lines in red i.e., Lines 4, 10, 17–29). For each training episode, DQN sets the initial state in Line 6. Next, for each time step within an episode, an action is chosen in an epsilon greedy manner (Lines 8–9) where the greedy action is the one with the maximal Q–value as approximated by the Q–function. The chosen action is executed and the outcome (immediate reward

and next state) is stored in the replay memory (Lines 11–12). Next, a minibatch of transitions (state, action, immediate reward, next state) is sampled and the Q–function weights (θ) are updated such that the squared error from the target Q–value, y_j, is reduced (Lines 13–15). The target Q–value (y_j) is computed using the temporal difference propagated from the next state [33]. Finally (Line 16), every C steps, the target Q–function ($\hat{Q}$) is set to equal the trained Q–function (Q). Doing such a delayed update is justified by Mnih et al. [23] as a way to reduce oscillations or divergence of the policy.

2.3 Related work

Previous works have examined reinforcement learning algorithms for online optimization of signal controllers. Unfortunately, the applicability of these protocols is questionable due to various drawbacks: **(a)** long and unsafe tuning process [9, 20, 25, 32, 36], **(b)** cumbersome policies that cannot be easily interpreted and regulated (mostly relying on deep artificial neural networks) thereby providing limited liability [9, 19, 20, 25, 32, 36, 40], **(c)** limited scalability (with regards to the number of managed phases) [19, 25, 36, 40], **(d)** reliance on (currently) unrealistic traffic sensing capabilities [25, 40], **(e)** experiments evaluated on unrealistic scenarios such as synthetic traffic demand [1, 4, 9, 19, 20, 25, 26], and handling only through traffic (no turning vehicles) [19, 25].

A line of previous work [3, 17, 26, 36, 39] focused on controlling and coordinating a set of signal controllers. The goal of such work is optimizing traffic flow over a road network that consists of several signalized intersections. Such a multiagent control manifests a combinatorial action space which results in limited scalability as well as slow and inefficient learning.

While some of the aforementioned publications presented compelling results, none yielded a control function that is human interpretable and liable. Given liability and regularity constraints faced by governmental transportation agencies, signal control functions will most likely not be adopted unless they can be interpreted and regulated.

Interpretability of deep reinforcement learning has recently been examined. Similar to this work, techniques for mimicking deep neural networks with interpretable actuation were suggested. One work [21] suggests using a variant of *continuous U trees*, however this approach requires that the affiliated deep Q–network be trained to convergence initially. As a result, the control policy remains uninterpretable for the duration of the training episodes. Consequently, such an approach cannot be utilized in safety critical domains that require online learning. This shortcoming is shared by other works [12, 37], each requiring learning a model that is based on a deep neural net prior to utilizing an interpretable controller.

3 PROBLEM DEFINITION

The focus of this paper is around online optimization of a human interpretable signal controller. Following previous work [27, 32, 40] we model this problem as an MDP. The state space is defined by the possible input assignments. The set of available actions per state is defined by the possible output assignments and constraints. Constraints are considered as part of the transition function of the environment. For example, actions do not include yellow lights, these are set automatically when required for safety. The transition

function is provided by the environment. Reward is defined by reduction in accumulated traffic delay. The controller's operation is defined as follows:

Input: (1) current signal assignment (green, yellow, and red assignment for each phase), for each lane: **(2)** number of approaching vehicles, **(3)** stopped vehicles accumulated waiting time, **(4)** number of stopped vehicles, and **(5)** average speed of approaching vehicles. Note that all inputs are necessarily non–negative.

Output: next signal assignment for a duration of time equal to a given minimum phase length. Signal assignments are abstracted to phases which group individual assignments into traffic movements. Non-conflicting pairs of phases give a complete signal assignment.

Constraints: (1) right–of–passage cannot be assigned to conflicting phases, **(2)** a yellow signal must appear for a predefined time interval between red and green signals.

Assumptions: (1) the intersection's layout is known, i.e., incoming/outgoing lanes, phases, and conflicting phases, **(2)** real–time sensing of incoming traffic as specified in the problem input. These sensing assumptions are reasonable given latest advances in traffic sensing technology, namely, radar [28], Wi–Fi scanning [15], drivers' smartphones [24], connected vehicles [38], image processing [5], infrared sensing [13], and Synthetic Aperture Radar (SAR) satellites [22].

Desiderata: The signal controller is expected to assign right–of–passage such that: **(1)** the average delay suffered by incoming vehicles is minimized **(2)** the actuation policy can be interpreted and regulated by a human (a precise definition is given next).

Regulatable signal controller policy

Control of safety critical tasks in general, and signal control specifically, require interpretable policies for liability and performance guarantee purposes. Unfortunately, there is little agreement on the meaning of interpretability. Ahmad et al [2018] state that "The choice of interpretable models depends upon the application and use case for which explanations are required". Consequently, this section provides a definition for an interpretable model for the signal domain.

A signal control policy is defined as, $\pi(s; \theta) \rightarrow (\Phi^g, \Phi^y, \Phi^r)$, mapping a given traffic state, s, and a set of parameters, θ, to a signal assignment, that is three sets of phases representing green, yellow, and red signal assignments, Φ^g, Φ^y, Φ^r, respectively. A valid signal assignment is one that results in no conflicting traffic movements. The traffic state, s, is defined by the provided sensors input. The control function's parameters, θ, should be tuned such that the control policy yields optimized performance.

We define a precedence function, $g(s, \Phi; \theta') \rightarrow \mathbb{R}$, mapping a given traffic state, s, a set of parameters, θ', and a non–conflicting set of signal phases, Φ, to a real number representing the precedence of assigning right–of–passage to Φ. Note that a Q-function [23] could serve as a precedence function but a precedence function is not necessarily a Q-function. Such a precedence function suggests a control policy where the chosen signal assignment in state s, is $\text{argmax}_\Phi g(s, \Phi; \theta')$. A series of precedence functions, G, one for each action, defines a full order over all phase assignments. θ' is tuned such that the action with the highest precedence is the optimal action in expectation.

DEFINITION 1 (REGULATABLE PRECEDENCE FUNCTION). *A precedence function g is defined as regulatable, if for all state variables* $s[i] \in s$, $\frac{\partial g}{\partial s[i]}$ *exists and is either non–negative or non–positive for any possible assignment of s, i.e., the precedence function is monotonic in the state variables.*

A control policy that is based on a regulatable precedence function is defined as a *regulatable control policy*. For such a policy, changes to the signal assignment can be intuitively interpreted as following changes in state variables, e.g., the right–of–passage was revoked from φ_2 and granted to φ_4 (as defined in Figure 1) because the number of stopped Southbound vehicles increased while the number of such Eastbound vehicles decreased. Moreover, should policy adjustment be required, adding a weighting parameter to each state variable allows for intuitive tuning of the control function with regards to the specific state variable.

Consider the example in Figure 2, traffic has accumulated on the Eastbound left–turn lane, however the light configuration has not yet been switched. A simple regulatable control function can be optimized which chooses between two configurations, allowing Westbound traffic, or allowing Eastbound traffic. In this example, the precedence function is a simple summation of 4 input variables (Queue length, number of approaching vehicles, accumulated waiting time for stopped vehicles, and average speed). Each of these variables is affiliated with one tunable parameter. E.g., "W-Through Queued" specifies the weight factor affiliated with the queue length variable for the Westbound trough lanes. We later generalize and discuss this type of controller in greater detail. The table at the bottom of the figure specifies optimized values for the different tunable parameters. By inspecting the values assigned we can observe that the speed of approaching vehicles on the Westbound left–turn lane is the primary factor in the decision to maintain the current light configuration. However, it might be the case that the Eastbound left–turn lanes are too short to accommodate the typical number of vehicles. The parameters can be easily tuned to give higher precedence to clearing traffic from these lanes by increasing the weight parameter E-Left Queued.
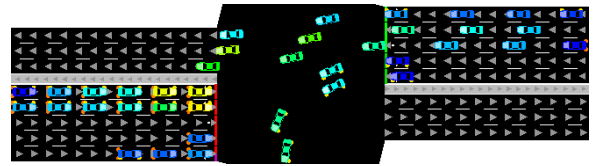
Action	Queued	Approaching	Waiting (s)	Speed	Value
W–Through	0.0	2.1	0.0	0.33	
W–Left	3.64	0.77	4.65	11.04	22.53
E–Through	1.25	0.0	0.73	0.0	
E–Left	5.32	1.67	10.34	0.0	19.31

Figure 2: Optimized actuation policy for the presented intersection (top). The policy is based on a linear control function that assigns values to input variables presented in the table (bottom).

This work makes the assumption that a regulatable control function is interpretable (a similar assumption was made in [12]).

4 DESIGNING A REGULATABLE PRECEDENCE FUNCTION

Based on the problem input (as defined in the problem definition) we define the following, phase dependent state variables, $s_\varphi[1, ..., 6]$: (1) number of stopped vehicles, (2) number of approaching vehicles, (3) cumulative stopped time, (4) average stopped time, (5) average queue length = stopped vehicles divided by the number of lanes, and (6) average speed for approaching vehicles. These phase dependent variables relate only to vehicles that are present on lanes served by phase φ. For example, s_{φ_1} returns the number of stopped vehicles on the Eastbound left–turning lanes. On top of these state variables, the proposed precedence function penalizes phase assignments that enforce clearance intervals. For instance, activating (assigning right–of–passage) the Southbound through phase straight after an active Eastbound through phase commonly requires a clearance interval. The clearance interval activation and duration is intersection dependent. For general guidelines see [7]. Transitioning from a currently active phase set to another phase set, Φ, triggers one of the following cases.

(1) **Full clearance** – an interval where no phase is active (all red).
(2) **Partial clearance** – an interval where part of the phases in Φ are inactive (prior to becoming active).
(3) **Permissive clearance** – a short interval (shorter than in the partial clearance case) where part of the phases in Φ are inactive. Commonly due to a currently active permissive–left phase (when applicable).
(4) **No clearance** – no clearance is required.

Each of these cases is affiliated with a flag, denoted $f_{1,...,4}$, that is set to 1 for the active case and 0 for the others. In a state where two cases are simultaneously active, only the one with the lowest index will be set to 1.

Our proposed precedence function can now be defined as:

$$g(s, \Phi; \theta') = \sum_{\varphi \in \Phi} \sum_{i=1}^{6} \left(w_i^{\Phi\varphi} s[i] \right)^{p_i^{\Phi\varphi}} \cdot \sum_{j=1}^{4} \left(w_j'^{\Phi} f_j \right)^{p_j'^{\Phi}} \quad (1)$$

Where $\theta' = \{w, p, w', p'\}$. In total, the designed function is composed of 6 state variables per φ in Φ, each with two affiliated tunable parameters, a weight ($w_{1...6}^{\Phi\varphi}$) and an exponent ($p_{1...6}^{\Phi\varphi}$). Also, each of the clearance flags ($f_{1...4}$) is affiliated with two tunable parameters, a weight ($w_{1...4}'^{\Phi}$) and an exponent ($p_{1...4}'^{\Phi}$). Each phase combination (Φ), phase (φ), and index (i or j) defines a unique tunable parameter. All together, this function defines 12 parameters per phase within a phase combination ($w^{\Phi\varphi}$, $p^{\Phi\varphi}$) and 8 parameters per phase combination (w'^{Φ}, p'^{Φ}). In the phase diagram defined in Figure 1 there are 8 phases which result in 8 sets of non–conflicting pairs. Namely, $\{(\varphi_1, \varphi_2) \times (\varphi_5, \varphi_6) \cup (\varphi_3, \varphi_4) \times (\varphi_7, \varphi_8)\}$. Consequently, the appropriate regulatable function G (over all 8 phase pair precedence functions) will be composed of $8(2 \cdot 12 + 8) = 256$ tunable parameters.

LEMMA 1. *The precedence function defined in Equation 1 is regulatable according to Definition 1.*

PROOF.

$$\frac{\partial g}{\partial s[i]} = w_i^{\Phi\varphi} p_i^{\Phi\varphi} s[i]^{p_i^{\Phi\varphi}-1} \cdot \sum_{j=1}^{4} \left(w_j'^{\Phi} f_j \right)^{p_j'^{\Phi}}$$

$s[i] \geq 0$ for any $i \in \{1, ..., 6\}$ (see 'input' in problem definition). Consequently, $s[i]^{p_i^{\Phi\varphi}-1} \geq 0$. All other components of the partial derivative i.e., $w_i^{\Phi\varphi} p_i^{\Phi\varphi} \cdot \sum_{j=1}^{4} \left(w_j'^{\Phi} f_j \right)^{p_j'^{\Phi}}$, are not dependent on any of the state variables and can be viewed as a constant (ϑ). As a result, $\frac{\partial g}{\partial s[i]}$ would be either non–negative ($\vartheta \geq 0$), non–positive ($\vartheta \leq 0$), or both ($\vartheta = 0$), for any i, s and a given Φ, θ' assignment. □

The precedence function defined in Equation 1 is hereafter denoted the *designed precedence function*. The affiliated control policy which returns $\text{argmax}_\Phi G(s, \Phi; \theta')$ is hereafter denoted the *designed control policy*. Next, we discuss general techniques for online tuning of θ' such that the performance of any regulatable control policy (and specifically the designed control policy) is optimized.

5 PARAMETER TUNING

A line of publications [20, 36, 40] reported that the DQN algorithm is particularly suitable for online signal control optimization. Unfortunately, the underlying policy in DQN is not regulatable as it is based on a DNN (Line 9 in Algorithm 1). In order to bridge this gap, we suggest training a given regulatable policy function of the type $\text{argmax}_a G(s, a; \widetilde{\theta})$, to imitate the Q–network actuation i.e., $\text{argmax}_a Q(s, a; \theta)$. A simple approach would be to directly use Q–learning with a function approximator that is defined as the regulatable function. However, doing so was found to make reaching a reasonable policy infeasible under even artificially low demands. Such a simple function approximator is incapable of representing the required intermediate functions for learning action values over an extended period of time.

Consequently, other approaches for leveraging DQN to optimize regulatable policies are considered. These approaches follow the pseudocode described in Algorithm 1. The lines in red (Lines 4, 10, 17–29) show the required additions on top of the original DQN algorithm. It is important to note that instead of selecting an action according to $\text{argmax}_a Q(s, a; \theta)$ (Line 9), the regulatable version selects an action according to a regulatable policy, $\text{argmax}_a G(s, a; \widetilde{\theta})$ where G is a previously initialized regulatable function (Line 4). Replacing the actuator in the DQN algorithm (Line 10 in lieu of Line 9) is reasonable as DQN is an *off–policy* algorithm, i.e., training the Q–function does not require that the same function interacts with the environment. Moreover, the powerful (DNN based) Q–function approximation that is trained by DQN can be used to train the regulatable function, G. Consequently, G is repeatedly trained to minimize the error between $\text{argmax}_a G$ and $\text{argmax}_a Q$. This training is performed at every time step over N random minibatches from the replay memory (Lines 17,18). Next, we discuss 3 different strategies for training G.

Algorithm 1: DQN and 3 DRQ variants shown in red

---

1   Initialize replay memory D to capacity N

2   Initialize Q function with random weights θ

3   Initialize target $\hat{Q}$ function with weights $\bar{\theta} = \theta$

4   Initialize regulatable function G with weights $\widetilde{\theta} = [1, ..., 1]$

5   **for** *episode* = 1 **to** M **do**

6   Initialize s_1 with observed state

7   **for** $t = 1$ **to** T **do**

8   With probability ϵ select a random action a_t
       otherwise select:

9       $a_t = \text{argmax}_a Q(s_t, a; \theta)$ # for DQN

10      $a_t = \text{argmax}_a G(s_t, a; \widetilde{\theta})$ # for regulatable

11   Execute action a_t and observe reward r_t and state
       s_{t+1}

12   Store transition (s_t, a_t, r_t, s_{t+1}) in D

13   Sample random minibatch of transitions
       (s_j, a_j, r_j, s_{j+1}) from D

14   Set $y_j = r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a'; \bar{\theta})$

15   Perform a gradient descent step on
       $(y_j - Q(s_j, a_j; \theta))^2$ with respect to θ

16   Every C steps reset $\hat{Q} = Q$

17   **for** $n = 1$ **to** N **do**

18   Sample random minibatch of transitions
         (s_j, a_j, r_j, s_{j+1}) from D

19   **case** *DRQ* **do**

20   Set $y_j = r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a'; \theta)$

21   Perform a gradient descent step on
          $(y_j - G(s_j, a_j; \widetilde{\theta}))^2$ with respect to $\widetilde{\theta}$

22   **case** *DRSQ* **do**

23   Set $X_j = \text{softmax}_a \left(Q(s_j, \cdot; \theta) \right)$

24   Set $Z_j = \text{softmax}_a \left(G(s_j, \cdot; \widetilde{\theta}) \right)$

25   Perform a gradient descent step on
          $-\sum_{a \in A} X_j[a] log(Z_j[a])$ w.r.t. $\widetilde{\theta}$

26   **case** *DRHQ* **do**

27   Set $X_j = \begin{cases} 1 & \text{argmax}_{a'} Q(s_j, a'; \theta) \\ 0 & a \in A \setminus \{a'\} \end{cases}$

28   Set $Z_j = \text{softmax}_a \left(G(s_j, \cdot; \widetilde{\theta}) \right)$

29   Perform a gradient descent step on
          $-\sum_{a \in A} X_j[a] log(Z_j[a])$ w.r.t. $\widetilde{\theta}$

5.1 Deep Regulatable Q–Learning (DRQ)

DRQ is our basic Regulatable Q–Learning variant where the parameters of the regulatable function $\widetilde{\theta}$ are tuned towards equivalency between G and the Q–function (using SGD with a squared loss function over the provided minibatch, Line 21). This variant follows the fact that if $G(s, a) = Q(s, a)$ for all (s, a) then the required policy equivalence is achieved i.e., $\forall s, \text{argmax}_a G(s, a) = \text{argmax}_a Q(s, a)$.

Attempting to tune $\widetilde{\theta}$ such that the regulatable function G would match the DNN–based Q–function may not be feasible in many cases. A regulatable function is more constrained and possesses far fewer tunable parameters compared to a DNN. As a result, a DNN is usually able to approximate a much larger set of functions compared to a regulatable approximator. Indeed, our empirical study found that the designed precedence function is very limited in its ability to approximate Q–values. However, setting the regulatable function, G, to imitate the action selection of DQN does not require that $G(s, a) \equiv Q(s, a)$. This understanding leads us to DRQ variants that provide extra flexibility with regards to the function approximated by G.

5.2 Deep Regulatable Softmax Q–Learning (DRSQ)

In DRSQ the parameters of the regulatable function $\widetilde{\theta}$ are tuned towards **proportional** equivalency between G and the Q–function. This variant follows the fact that if $G(s, \cdot) \propto Q(s, \cdot)$ for all s then the required policy equivalence is achieved i.e., $\forall s, \text{argmax}_a G(s, a) = \text{argmax}_a Q(s, a)$. The proportionality values over all actions are standardized using the softmax function. As a result the SGD applied for tuning $\widetilde{\theta}$ takes a gradient based on the cross–entropy objective i.e., using the log loss function (Lines 22–25).

5.3 Deep Regulatable Hardmax Q–Learning (DRHQ)

DRHQ stems from the understanding that policy equivalency does not require full equivalency or even proportional equivalency between G and the Q–function. In fact, for achieving policy equivalency it is sufficient to tune $\widetilde{\theta}$ directly towards argmax_a equivalency between G and the Q–function. This can be achieved by setting the target value for a given s, a pair as 1 for $\text{argmax}_a Q(s, a)$ or 0 otherwise (Line 27). Next, SGD is used to tune $\widetilde{\theta}$ according to the log loss between the target values and softmax $(G(s, \cdot))$.

5.4 Other tuning approaches

The covariance matrix adaptation evolutionary strategy (CMA–ES) algorithm [11] is known to be a particularly effective parameter tuning approach. CMA–ES is specifically suitable for domains where the tunable parameters have a continuous value range. Moreover, CMA–ES is known for having few hyper–parameters with fairly low sensitivity. As a result, it is particularly appealing for testing and validating our designed policy. On the other hand, CMA–ES may be unsuitable for online tuning due to inefficient data sampling, requiring several full episodes for a single policy update step. Moreover, the erratic exploration of CMA–ES, though helpful in avoiding local optimums, is less suitable for safety critical domains.

Another natural candidate for online parameter tuning is the policy gradient approach [34]. Specifically, the Proximal–Policy Optimization (PPO) algorithm [31] is particularly suitable for safety–critical domains as it encourages bounded policy gradient steps. PPO achieves this behaviour by clipping the expected advantage for large policy divergence. Consequently, PPO is expected to result in smooth and steady convergence but, on the other hand, is more prone to settle in a local optimum and achieve sub–optimal performance.

Demand	Total	Avg Rate	Low Rate	High Rate
Low	45,112	1.04	0.59	1.29
Medium	51,298	1.19	0.76	1.42
High	61,261	1.42	0.98	1.59

Table 1: Traffic demand for three different days representing low, medium, and high traffic volumes. Each day is affiliated with: the total number of approaching vehicles (Total), approaching vehicle per second on average (Avg Rate), during off–peak (Low Rate), and during peak hour (High Rate).

6 EMPIRICAL STUDY

The purpose of the empirical study is to evaluate and analyze the performance of the proposed designed control policy along with the affiliated online tuning algorithms. Specifically, the empirical study aims at answering the following questions:

(1) Can the designed control policy approximate a deep Q–learning optimized policy?
(2) Is a policy gradient optimization approach suitable for training the designed control policy?
(3) How do the regulatable Q–learning variants compare to the state–of–the–art, DQN–based signal controller?

6.1 Experimental settings

The reported experiments rely on a well–established traffic simulator, Simulation of Urban MObility (SUMO) [6], along with traffic scenarios that are based on real–life observations. The Utah department of transportation (UDOT) provides an open access database (https://udottraffic.utah.gov/ATSPM) specifying the observed traffic demand for 2092 signalized intersections. The database specifies the number of vehicles affiliated with each incoming, outgoing road combination in 5 minutes aggregation. The demand reported by UDOT is parsed into SUMO where vehicles are spawned with equal probability along equivalent 5 minute intervals.

The reported results relate to a representative major intersection, State St & E 4500 S, Murray, Utah. Source code for all experiments is available at: *https://github.com/jault/StateStreetSumo*. This intersection is chosen as it is affiliated with high volumes of traffic arriving from two arterial roads. It typically receives more than 50,000 vehicles a day, peaking at 95 cars per minute in rush hour. Figure 3 provides a snapshot of the simulated intersection (right) and a picture from the actual intersection (left). This intersection is affiliated with 10 phases, the 8 reported in Figure 1, and 2 North and South bound permissive left turns. The minimum phase length is set to 3 seconds for this intersection. These 10 phases form 11 unique pairs of non–conflicting phase combinations. As a result, the affiliated designed control policy has 352 tunable parameters. The UDOT database (Signal #7157) specifies the affiliated clearance interval's activation and duration.

In order to examine various traffic conditions, demand from 3 different days is chosen for simulation, Wednesday – May 1, Monday – May 6, and Friday – June 21, all of 2019. For each day, a 14 hour time period is considered from 7 A.M. to 9 P.M. These dates were chosen as representative examples of low (May 6th), medium (May 1st), and high (June 21st) traffic volumes. Table 1 presents

the total number of vehicles that crossed the intersection on each day as well as the average number of vehicles arriving per second, during off–peak (Low Rate), and peak–hour (High Rate).

The reported CMA–ES implementation is based on pycma [10], and the initial variance and population size were chosen to be 0.2 and 12 respectively. The simulated intersection controller is defined as an environment within OpenAI's GYM [8], which gives a standard interface for reinforcement learning. Hyper–parameters for the online algorithms under all demand profiles aside from discount factor are identical throughout the different variants. The discount factor and epsilon were chosen empirically. Other values resulted in similar trends yet yielded slightly worse outcomes. Under low and medium demand the discount factor is set as 0.8. The discount is raised to 0.9 in high demands; this change was found to be important as clearing traffic from a set of lanes requires a larger planning horizon.

For DQN and the regulatable variants, the minibatch size is set to 32 and replay capacity at 100,000 transitions. The epsilon greedy action–selection probability in DQN and its variants is reduced from 0.05 to 0 (full exploit) after 20 training episodes resulting in the observed drop in the graphs. The Q–network is a DNN composed of 3 hidden layers with 64 units each, where the first layer is a 2x2 kernel convolutional layer grouping the input for lanes that belong to the same road. The Huber loss function is used in line with the original DQN work. Leaky ReLU activation is used for all layers along with the Adam optimizer with a step–size of 0.001, and decay rates β_1 and β_2 are set as 0.9 and 0.999 respectively. The same parameters were used for both optimizers in the two–stage regulatable variants. The PPO implementation follows the advantage actor–critic paradigm defining the actor by the regulatable function and the critic by a similar neural network as described prior with an alternative objective of estimating state advantages. The step size for each optimizer in this case are 0.001 for the actor and 0.001 for the critic with all other hyper parameters remaining the same.

Finally, as a baseline for comparison, the results include performance measures for the commonly deployed actuated signal control [7]. This type of controller is provided by the SUMO simulator. Phases for the actuated controller are set in a fixed order, protected lefts followed by through traffic. The maximum green time is set to 300 seconds.

6.2 Regulatable control function

The first set of experiments aim at addressing the question: can the designed control policy approximate a deep Q–learning optimized policy?

Given the stochastic nature of the signal domain and the combinatorial state space, it is infeasible to compute the true optimal performance. Instead, we say that the control function can approximate an optimized policy if it results in performance that is competitive with a state–of–the–art, DNN based controller [32]. In order to address this question, a parameter tuning process is applied to the regulatable function. CMA–ES is chosen as the optimization approach since it is comparatively insensitive to hyper–parameter settings with just the population size and initial variance as hyper–parameters. Further, policy performance can be determined over

Figure 3: A signalized intersection at State St & E 4500 S, Murray, Utah (left), (picture credit: Google Maps), and an equivalent intersection modeled in SUMO (right).
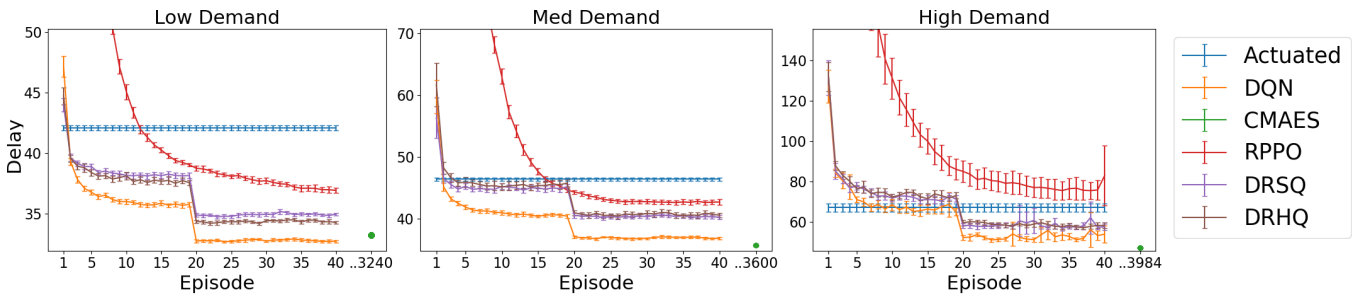

Figure 4: Average seconds of delay caused by the intersection under each demand profile. In each experiment the 95% confidence intervals per episode over a population of 30 trials with random seeds is given.

entire episodes as cma–es is highly parallelizable. Finally, the simulator provides our goal of delay directly when vehicles exit the simulation. This allows for optimization of the designed control policy with respect to the delay rather than approximated through the observed waiting time as real–time algorithms require.

Figure 4 presents the average delay suffered by approaching vehicles for each of the three representative days as a function of the training episode (full day of traffic). For now, consider the DQN curve (representing the state–of–the–art, DNN based controller) and the cma–es datapoint (in green). As can be seen, cma–es with the designed regulatable control function achieves competitive delay measurements on all three traffic scenarios. Note that, cma–es is presented as a single data–point (post convergence), as it requires two orders of magnitude more samples in order to converge compared to the other approaches. The full tuning process cannot be fitted on the presented plots.

It is important to note that other, simpler regulatable control functions were also examined, specifically based on Polynomial [16], and Fourier basis [14] function approximators. Both yielded extremely poor performance. After over 100 epochs either policy failed to complete the scenario by clearing all vehicles by the end of the simulated time period. The polynomial function showed some improvement, while Fourier basis was stagnant.

Despite its impressive ability to optimize the designed, regulatable function, cma–es is not practical for online optimization for two reasons. (1) Inefficient sampling – each learning step (update of parameters' mean value) requires 24 episodes (24 full days of traffic). (2) Erratic exploration – extreme parameters values are sampled leading to unacceptable performance during the tuning process. These drawbacks are immediately evident in Figure 5 where the average delay over each epoch is presented. cma–es fails to meet actuated performance for at least 25 epochs (600 episodes) for the medium demand scenario (similar trends were observed for the high and low demand scenarios). The erratic exploration is a hindrance as some solutions entirely fail to complete the scenario or perform much worse than average. Performance doesn't become stable for nearly another 4,000 episodes (11 years of traffic). Nonetheless, these results are still valuable as they provide a promising lower bound estimate on the performance of the regulatable policy (as seen in Figure 4).

6.3 Policy gradient approach

Policy–gradient approaches [41] are known to be especially suitable for optimizing a given policy function while providing some bounds on the exploration rate [30, 31]. As a result, such algorithms are promising candidates for online tuning of regulatable functions in safety critical domains.

Figure 4 presents the performance curve for tuning the designed control policy using the PPO algorithm. As expected, the learning
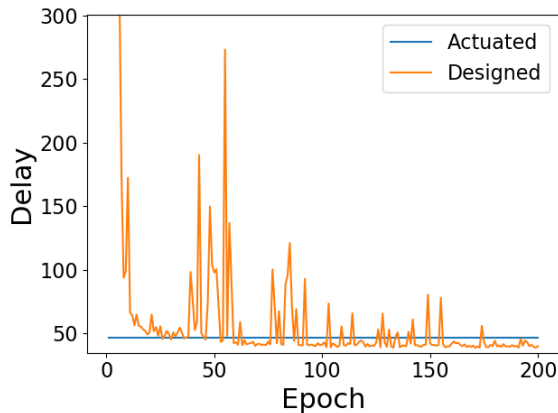
Figure 5: Average seconds of delay caused by the intersection under medium delay optimized by CMA–ES plotted as the mean of 24 episode epochs.

curve is smooth and monotonic suggesting a safe exploration process. On the other hand, PPO requires about 15 episodes to reach actuated control performance in low and medium demand, while not even reaching actuated level performance in the case of high demand. As a result, such an approach is unlikely to be adopted in practice. Moreover, bounded exploration can cause PPO to converge to suboptimal local optimums as evidenced by the presented results.

6.4 Q–learning

Given that the DQN algorithm achieves state–of–the–art results in the signal control domain (yellow curve in Figure 4), the next set of experiments is set to examine how the regulatable Q–learning variants compare.

Figure 4 presents the performance curve for DRSQ (in purple) and DRHQ (in brown). As stated above, vanilla DRQ performs significantly worse and is, thus, not presented in this plot. Both DRSQ and DRHQ outperform PPO and, for low and medium demand, outperform actuated control after a single episode. On the other hand, it takes them 20 episodes to outperform actuated control for high demand (once epsilon is reduced to 0). In order to mitigate the long training time on high demand, Laval et al. [2019] suggested to train the controller on low demand prior to applying it to high demand scenarios. Such an approach is expected to be specifically useful for training regulatable control functions.

DRHQ has a small, yet significant, advantage over DRSQ for low demand while yielding similar performance for medium and high demand. Finally, while DQN outperforms both, it is important to keep in mind that the policy induced by DQN cannot be interpreted and regulated. Consequently, DQN is unlikely to be implemented and serves only as an expected lower bound on regulatable performance.

7 SUMMARY AND CONCLUSIONS

This paper discusses online optimization of interpretable signal control policies. Unlike previous work that based such controllers

on deep neural networks, this paper suggests utilizing a policy function that can be interpreted and regulated. A regulatable policy function is defined as one with a monotonic relation between each state variable and the precedence of a given action. The following conclusions are drawn.

1. A regulatable function can approximate an optimized policy in a way that is competitive with a deep neural network.

2. A policy gradient approach is not suitable for training a regulatable function in this domain due to slow and sub–optimal convergence.

3. A Q–learning approach which trains a regulatable function results in good performance both with regards to convergence speed and the final policy. The regulatable function should be trained to fit the hardmax action as provided by the underlying deep Q–network according to DRHQ.

Future work will examine techniques for warm starting the learning process by observing the operation of a currently deployed controller.

REFERENCES

[1] Monireh Abdoos, Nasser Mozayani, and Ana LC Bazzan. 2011. Traffic light control in non-stationary environments based on multi agent Q-learning. In *2011 14th International IEEE conference on intelligent transportation systems (ITSC)*. IEEE, 1580–1585.

[2] Muhammad Aurangzeb Ahmad, Carly Eckert, and Ankur Teredesai. 2018. Interpretable machine learning in healthcare. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*. ACM, 559–560.

[3] Mohammad Aslani, Stefan Seipel, Mohammad Saadi Mesgari, and Marco Wiering. 2018. Traffic signal optimization through discrete and continuous reinforcement learning with robustness analysis in downtown Tehran. *Advanced Engineering Informatics* 38 (2018), 639–655.

[4] Bram Bakker, Shimon Whiteson, Leon Kester, and Frans CA Groen. 2010. Traffic light control by multiagent reinforcement learning systems. In *Interactive Collaborative Information Systems*. Springer, 475–510.

[5] Carlo Migel Bautista, Clifford Austin Dy, Miguel Iñigo Mañalac, Raphael Angelo Orbe, and Macario Cordel. 2016. Convolutional neural network for vehicle detection in low resolution traffic videos. In *2016 IEEE Region 10 Symposium (TENSYMP)*. IEEE, 277–281.

[6] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. 2011. SUMO–simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind.

[7] James Bonneson, Srinivasa R Sunkari, Michael Pratt, Praprut Songchitruksa, et al. 2011. *Traffic signal operations handbook*. Technical Report. Texas. Dept. of Transportation. Research and Technology Implementation Office.

[8] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. (2016). arXiv:arXiv:1606.01540

[9] Wade Genders and Saiedeh Razavi. 2016. Using a deep reinforcement learning agent for traffic signal control. *arXiv preprint arXiv:1611.01142* (2016).

[10] Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. 2019. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634. (Feb. 2019). https://doi.org/10.5281/zenodo.2559634

[11] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation* 9, 2 (2001), 159–195.

[12] Daniel Hein, Steffen Udluft, and Thomas A Runkler. 2018. Interpretable policies for reinforcement learning by genetic programming. *Engineering Applications of Artificial Intelligence* 76 (2018), 158–169.

[13] Yoichiro Iwasaki, Shinya Kawata, and Toshiyuki Nakamiya. 2011. Robust vehicle detection even in poor visibility conditions using infrared thermal images and its application to road traffic flow monitoring. *Measurement Science and Technology* 22, 8 (2011), 085501.

[14] George Konidaris, Sarah Osentoski, and Philip Thomas. 2011. Value function approximation in reinforcement learning using the Fourier basis. In *Twenty-fifth AAAI conference on artificial intelligence*.

[15] Vassilis Kostakos, Timo Ojala, and Tomi Juntunen. 2013. Traffic in the smart city: Exploring city-wide sensing for traffic control center augmentation. *IEEE Internet Computing* 17, 6 (2013), 22–29.

[16] Michail G Lagoudakis and Ronald Parr. 2003. Least-squares policy iteration. *Journal of machine learning research* 4, Dec (2003), 1107–1149.

[17] Jorge A Laval and Hao Zhou. 2019. Large-scale traffic signal control using machine learning: some traffic flow considerations. *arXiv preprint arXiv:1908.02673* (2019).

[18] David M Levinson. 1998. Speed and delay on signalized arterials. *Journal of Transportation Engineering* 124, 3 (1998), 258–263.

[19] Li Li, Yisheng Lv, and Fei-Yue Wang. 2016. Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica* 3, 3 (2016), 247–254.

[20] Xiaoyuan Liang, Xunsheng Du, Guiling Wang, and Zhu Han. 2018. Deep reinforcement learning for traffic light control in vehicular networks. *arXiv preprint arXiv:1803.11115* (2018).

[21] Guiliang Liu, Oliver Schulte, Wang Zhu, and Qingcan Li. 2018. Toward interpretable deep reinforcement learning with linear model u-trees. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 414–429.

[22] Franz Meyer, Stefan Hinz, Andreas Laika, Diana Weihing, and Richard Bamler. 2006. Performance analysis of the TerraSAR-X traffic monitoring concept. *ISPRS Journal of Photogrammetry and Remote Sensing* 61, 3-4 (2006), 225–242.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[24] Prashanth Mohan, Venkata N Padmanabhan, and Ramachandran Ramjee. 2008. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, 323–336.

[25] Seyed Sajad Mousavi, Michael Schukat, and Enda Howley. 2017. Traffic light control using deep policy-gradient and value-function-based reinforcement learning. *IET Intelligent Transport Systems* 11, 7 (2017), 417–423.

[26] K. J. Prabuchandran, Kumar A. N. Hemanth, and Shalabh Bhatnagar. 2014. Multi-agent reinforcement learning for traffic signal control. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2529–2534.

[27] LA Prashanth and Shalabh Bhatnagar. 2011. Reinforcement learning with function approximation for traffic signal control. *IEEE Transactions on Intelligent Transportation Systems* 12, 2 (2011), 412–421.

[28] P Samczynski, K Kulpa, M Malanowski, P Krysik, et al. 2011. A concept of GSM-based passive radar for vehicle traffic monitoring. In *2011 MICROWAVES, RADAR AND REMOTE SENSING SYMPOSIUM*. IEEE, 271–274.

[29] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. 2017. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296* (2017).

[30] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International Conference on Machine Learning*. 1889–1897.

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[32] Soheil Mohamad Alizadeh Shabestary and Baher Abdulhai. 2018. Deep Learning vs. Discrete Reinforcement Learning for Adaptive Traffic Signal Control. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 286–293.

[33] Satinder P Singh and Richard S Sutton. 1996. Reinforcement learning with replacing eligibility traces. *Machine learning* 22, 1-3 (1996), 123–158.

[34] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[35] Alejandro Tirachini. 2013. Estimation of travel time and the benefits of upgrading the fare payment technology in urban bus services. *Transportation Research Part C: Emerging Technologies* 30 (2013), 239–256.

[36] Elise Van der Pol and Frans A Oliehoek. 2016. Coordinated deep reinforcement learners for traffic light control. *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)* (2016).

[37] Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. 2018. Programmatically interpretable reinforcement learning. *arXiv preprint arXiv:1804.02477* (2018).

[38] Jiafu Wan, Jianqi Liu, Zehui Shao, Athanasios Vasilakos, Muhammad Imran, and Keliang Zhou. 2016. Mobile crowd sensing for traffic prediction in internet of vehicles. *Sensors* 16, 1 (2016), 88.

[39] Xiaoqiang Wang, Liangjun Ke, Zhimin Qiao, and Xinghua Chai. 2019. Large-scale Traffic Signal Control Using a Novel Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:1908.03761* (2019).

[40] Hua Wei, Guanjie Zheng, Huaxiu Yao, and Zhenhui Li. 2018. Intellilight: A reinforcement learning approach for intelligent traffic light control. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2496–2505.

[41] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.