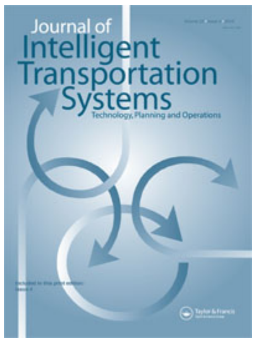

Journal of Intelligent Transportation Systems
Technology, Planning, and Operations

ISSN: 1547-2450 (Print) 1547-2442 (Online) Journal homepage: https://www.tandfonline.com/loi/gits20

Asynchronous *n*-step Q-learning adaptive traffic signal control

Wade Genders & Saiedeh Razavi

To cite this article: Wade Genders & Saiedeh Razavi (2019) Asynchronous *n*-step Q-learning adaptive traffic signal control, Journal of Intelligent Transportation Systems, 23:4, 319-331, DOI: 10.1080/15472450.2018.1491003

To link to this article: https://doi.org/10.1080/15472450.2018.1491003

Published online: 03 Jan 2019.

JOURNAL OF INTELLIGENT TRANSPORTATION SYSTEMS
2019, VOL. 23, NO. 4, 319–331
https://doi.org/10.1080/15472450.2018.1491003

Asynchronous *n*-step Q-learning adaptive traffic signal control

Wade Genders and Saiedeh Razavi

Department of Civil Engineering, McMaster University, Hamilton, Ontario, Canada

ABSTRACT
Ensuring transportation systems are efficient is a priority for modern society. Intersection traffic signal control can be modeled as a sequential decision-making problem. To learn how to make the best decisions, we apply reinforcement learning techniques with function approximation to train an adaptive traffic signal controller. We use the asynchronous *n*-step Q-learning algorithm with a two hidden layer artificial neural network as our reinforcement learning agent. A dynamic, stochastic rush hour simulation is developed to test the agent's performance. Compared against traditional loop detector actuated and linear Q-learning traffic signal control methods, our reinforcement learning model develops a superior control policy, reducing mean total delay by up 40% without compromising throughput. However, we find our proposed model slightly increases delay for left turning vehicles compared to the actuated controller, as a consequence of the reward function, highlighting the need for an appropriate reward function which truly develops the desired policy.

ARTICLE HISTORY
Received 21 October 2017
Revised 13 June 2018
Accepted 16 June 2018

KEYWORDS
Artificial intelligence; intelligent transportation systems; neural networks; reinforcement learning; traffic signal controllers

1. Introduction

Society relies on its many transportation systems for the movement of individuals, goods, and services. Ensuring vehicles can move efficiently from their origin to destination is desirable by all. However, increasing population, and subsequent vehicle ownership, has increased the demand of road infrastructure often beyond its capacity, resulting in congestion, travel delays, and unnecessary vehicle emissions. To address this problem, two types of solutions are possible. The first is to increase capacity by expanding road infrastructure; however, this can be expensive, protracted, and decrease capacity in the short term (i.e. parts of the existing infrastructure are often restricted to traffic while construction occurs). The second solution is to increase the efficiency of existing infrastructure and the systems that govern them, such as traffic signal controllers (TSC). We advocate this second solution, by utilizing innovations from the domain of artificial intelligence (Mnih et al., 2015, 2016) to develop an adaptive traffic signal controller.

Developing a traffic signal controller means attempting to solve the traffic signal control problem (i.e. deciding what traffic phases should be enacted at any given time). Many solutions have been proposed to solve the traffic signal control problem, differing in complexity and assumptions made. The traffic signal control problem can be modeled as a sequential decision-making problem. Reinforcement learning, a machine learning paradigm, provides a framework for developing solutions to sequential decision-making problems. A reinforcement learning agent learns how to act (i.e. control the intersection phases) in an uncertain environment (i.e. stochastic vehicle dynamics) to achieve a goal (i.e. maximize a numerical reward signal, such as throughput). Rewards r are quantitative measures of how well the agent is achieving its goal. Described mathematically, given a discount factor $\gamma \in [0, 1)$, the agent seeks to maximize the sum of discounted future rewards $G_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$. Formally, the agent develops a policy π which maps states to actions $\pi(s) = a$, in an attempt to maximize the sum of discounted future rewards. To achieve optimal control, we develop an optimal state-action policy π^*. We use valued-based reinforcement learning to estimate an action-value function, which tells us how "good" any action is in a given state based on the agent's prior experiences.

CONTACT Wade Genders genderwt@mcmaster.ca Department of Civil Engineering, McMaster University, Hamilton, Ontario, Canada.
Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/gits.

We derive the optimal policy by developing an optimal action-value function $Q^*(s, a)$ defined in Equation (1).

$$Q^*(\mathrm{s}, \mathrm{a}) = \max_{\pi} \mathbb{E}[G_t | s = s_t, \ a = a_t, \pi] \tag{1}$$

Q-learning (Watkins & Dayan, 1992), a type of temporal difference learning (Sutton, 1988), is used to estimate the optimal action-value function through repeated environment interactions, composed of the agent, at time t, observing *the state of the environment* s_t, choosing an action a_t, receiving feedback in the form of a reward r_t and observing a new state s_{t+1}. Accurately estimating an action's value allows the agent to select the action which will yield the most long-term reward.

However, traditional reinforcement learning can suffer from the *curse of dimensionality*, where developing tabular solutions becomes intractable due to a combinatorial explosion of the state-action space. This problem can be alleviated by combining traditional reinforcement learning with function approximators and supervised learning techniques. We approximate the optimal action-value function using an artificial neural network, defined in Equation (2).

$$Q^*(s, a) = Q^*(s, a; \theta) \tag{2}$$

We develop the parameters θ (weights between neurons) of the neural network action-value function using a variant of the deep Q-network (DQN) algorithm (Mnih et al., 2015), asynchronous n-step Q-learning (Mnih et al., 2016). Once sufficiently approximated, the optimal policy can be derived by selecting the action with the highest value using the developed action-value function, defined in Equation (3).

$$\pi^*(s) = \operatorname*{argmax}_{a} Q^*(s, a; \theta) \tag{3}$$

Using the microtraffic simulator SUMO (Krajzewicz, Erdmann, Behrisch, & Bieker, 2012) as the environment, we ultimately produce an agent that can control the traffic phases, termed an n-step Q-network traffic signal controller (nQN-TSC), illustrated in Figure 1.

Other researchers have long noted reinforcement learning's potential in solving the traffic signal control problem and have applied it with varying degrees of success (Balaji, German, & Srinivasan, 2010; El-Tantawy, Abdulhai, & Abdelgawad, 2013; Prashanth & Bhatnagar, 2011). Recent deep reinforcement learning techniques, successful in other domains, have provided further improvements over traditional reinforcement learning, traffic signal control models (Casas, 2017; Li, Lv, & Wang, 2016; Mousavi, Schukat, Corcoran, & Howley, 2017; Rijken, 2015; van der Pol, 2016). However, we observed subtle deficiencies in previous work which we address in this paper; specifically, the traffic signal control problem has been modeled too simply (discussed in Section 2. Literature review) and must be confronted authentically. We argue these simplifications have created uncertainty as to whether reinforcement learning can truly be a contending solution for the traffic signal control problem compared against established solutions.

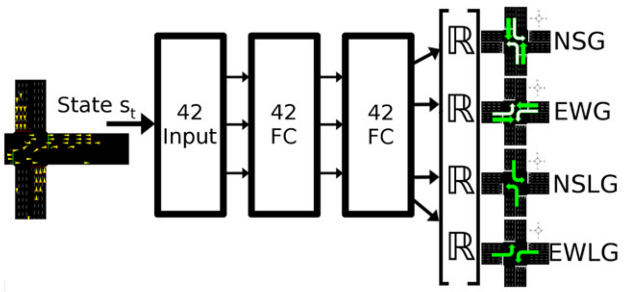

Figure 1. nQN-TSC model. First, the density and queue state s_t is observed by the agent, which is used as input to the neural network [Fully Connected (FC) layer of 42 neurons connected to another FC layer of 42 neurons], which produces a column vector of real numbers, each representing a different actions' value. Solid green arrows represent protected movements and white arrows represent permissive movements.

2. Literature review

Intersection traffic signal control methods have been studied for decades. Traditionally, traffic signal phases are displayed in an ordered sequence to produce a cycle. If the phase durations and sequence are static, a fixed cycle traffic signal controller is produced and can be classified as a nonadaptive form of traffic signal control. Nonadaptive signal control methods are stable and relatively simple to create, yet may not be optimal as they are not influenced by current traffic conditions. With the advent of sensors (e.g. loop detectors, radar, cameras) at intersections, adaptive traffic signal controllers were possible. Adaptive traffic signal controllers can vary phase duration or phase sequence, producing dynamic phase durations and acyclic phase sequences. Adaptive methods can potentially improve performance as they react to current traffic conditions, but can be unstable, complex, and difficult to develop.

We propose reinforcement learning as the appropriate method for traffic signal control because it offers specific advantages compared to other solutions. First, reinforcement learning utilizes the structure of the considered problem, observing what actions in certain states achieve reward (Sutton & Barto, 1998).

This is in contrast to other optimization techniques, such as evolutionary methods (e.g. genetic algorithms; Lee, Abdulhai, Shalaby, & Chung, 2005), which ignore the information yielded from individual environment interactions. Second, Q-learning is a model-free reinforcement learning algorithm, requiring no model of the environment once it has been sufficiently trained. Contrasted with many traffic signal control optimization systems (e.g. SCOOT – Hunt, Robertson, Bretherton, & Winton, 1981; SCATS – Lowrie, 1990; RHODES – Mirchandani & Head, 2001) which require traffic models, or control theory approaches (Gregoire, Qian, Frazzoli, De La Fortelle, & Wongpiromsarn, 2015; Timotheou, Panayiotou, & Polycarpou, 2015; Wongpiromsarn, Uthaicharoenpong, Wang, Frazzoli, & Wang, 2012) which use other models, such as backpressure, model-free reinforcement learning makes no assumptions about the model as no model is required. This arguably makes reinforcement learning more robust to traffic dynamics; model-based methods require their models accurately reflect reality. As the disparity between the model and reality increases, the performance of the system suffers. Model-free reinforcement learning is parsimonious compared to model-based methods, requiring less information to function.

Significant research has been conducted using reinforcement learning for traffic signal control. Early efforts were limited by simple simulations and a lack of computational power (Abdulhai, Pringle, & Karakoulas, 2003; Brockfeld, Barlovic, Schadschneider, & Schreckenberg, 2001; Thorpe & Anderson, 1996; Wiering, 2000). Beginning in the early 2000s, continuous improvements in both of these areas have created a variety of simulation tools that are increasingly complex and realistic. Traffic microsimulators are the most popular tool used by traffic researchers, as they model individual vehicles as distinct entities and can reproduce real-world traffic behavior such as shockwaves. Research conducted has differed in reinforcement learning type, state space definition, action space definition, reward definition, simulator, traffic network geometry and vehicle generation model, and can be separated into traditional reinforcement learning and deep reinforcement learning. Previous traditional reinforcement learning traffic signal control research efforts have defined the state space as an aggregate traffic statistic, the number of queued vehicles (Abdoos, Mozayani, & Bazzan, 2013; Abdulhai et al., 2003; Aziz, Zhu, & Ukkusuri, 2018; Chin, Bolong, Kiring, Yang, & Teo, 2011; Wiering, 2000) and traffic flow (Arel, Liu, Urbanik, & Kohls, 2010; Balaji et al., 2010) the most popular. The action space has been defined as all available signal phases (Arel et al., 2010; El-Tantawy et al., 2013) or restricted to green phases only (Abdoos et al., 2013; Balaji et al., 2010; Chin et al., 2011). The most common reward definitions are functions of delay (Arel et al., 2010; El-Tantawy et al., 2013) and queued vehicles (Abdoos et al., 2013; Balaji et al., 2010; Chin et al., 2011). For a comprehensive review of traditional reinforcement learning traffic signal control research, the reader is referred to (El-Tantawy, Abdulhai, & Abdelgawad, 2014) and (Mannion, Duggan, & Howley, 2016). Bayesian reinforcement learning has also been demonstrated effective for adaptive traffic signal control (Khamis & Gomaa, 2012, 2014; Khamis, Gomaa, El-Mahdy, & Shoukry, 2012; Khamis, Gomaa, & El-Shishiny, 2012).

Deep reinforcement learning techniques were first applied to traffic signal control via the DQN in (Rijken, 2015). The authors followed the example in (Mnih et al., 2015) closely, defining the dimensions of the state space to be the same dimensions as in the Atari environment, meaning only a fraction of any state observation contained relevant information (which the authors identified and acknowledged as problematic). A uniform distribution was used to generate vehicles into the network, which is likely too simple a model of vehicles arriving at an intersection. Also, only green phases were modeled, meaning no yellow change or red clearance phases were used between green phases.

Subsequent work by (van der Pol, 2016) expanded on the ideas in (Rijken, 2015), with the inclusion of vehicle speed and signal phases in the state observation and 1 s yellow change phases, Double Q-learning and limited experiments with multiple intersections. Both authors noted that while the agent's performance improved with training, there were concerns with the agent's learned policy, such as quickly oscillating phases and the inability to develop stable, long-term policies.

Research by (Li et al., 2016) deviated from previous work using deep stacked autoencoders to approximate the optimal action-value function. A disadvantage of stacked autoencoders is that each layer must be trained individually until convergence and then stacked on top one another, as opposed to DQN, which can be trained quicker in an end-to-end manner. The state observation was the number of queued vehicles in each lane over the past 4 s. The authors tested their agent on an isolated four way, two lane intersection where vehicles could not turn left or right, only travel through. This simplified the action space to two green phases. They also mandated a minimum green time of 15 s and did not model any yellow change or red clearance phases. The traffic demand for each intersection approach was [100, 2000] veh/h generated "randomly," without detailing the specific probability distribution.

Research referenced thus far used valued-based reinforcement learning, which estimate value functions to develop the traffic signal control policy. Policy-based methods are another type of reinforcement learning which explicitly parameterize the policy instead of using a value function to derive the policy. A traffic signal control policy was parameterized in (Casas, 2017) and (Mousavi et al., 2017). Deep policy gradient was used in (Casas, 2017) to control dozens of traffic signal controllers in a large traffic network with impressive results. However, the model developed modified an existing traffic signal control policy instead of deriving its own in a truly autonomous manner. It seems this was a concession made so that multiple intersections could be controlled with the same agent; however, it makes the implicit assumption that the optimal policy for each intersection is the same, which is unlikely. Regardless, the size and scope of the experiments which the agent was subjected, and its subsequent performance, is strong evidence for further work using policy-based reinforcement learning for traffic signal control.

3. Contribution

Considering the previous work as detailed, we identify the following problems and our contributions.

First, prior research has modeled the traffic signal problem with unrealistic simplifications. Specifically, either completely omitting yellow change and red clearance phases (Li et al., 2016; Rijken, 2015), or including them for short durations (van der Pol, 2016). Acknowledging that yellow and red phase durations can depend on many factors (e.g. intersection geometry and traffic laws), we contend that omitting yellow and red phases or including them for unrealistically short durations simplifies the traffic signal control problem such that any results are questionable. If reinforcement learning methods are to provide a solution to the traffic signal control problem, the modeling must include all relevant aspects as would be present in practice. We contribute a realistic model of the traffic signal control problem, with yellow and red phase durations of 4 s. Although the inclusion of these additional signals may seem slight, it changes how the agent's actions are implemented.

Second, we contend the only actions worth choosing by the agent are the green phases, as these are the actions which ultimately achieve any rational traffic signal control goal (i.e. maximize throughput, minimize delay or queue), assuming safety concerns have been satisfied. However, if green phases are the only actions available to the agent, depending on the current phase, some actions cannot always be enacted immediately – yellow change and red clearance phases may be necessary. For example, there exist action sequences which are unsafe to transition from immediately, meaning the traffic signal phase cannot immediately change to the agent's chosen action for safety reasons (i.e. the movements constituting these phases conflict). Some action sequences may require the traffic signal to enact a yellow change phase, instructing vehicles to slow down and prepare to stop, and then a red clearance phase, instructing vehicles to stop. Note that the agent does not explicitly select these transition traffic phases as actions, they are implicit when selecting actions that conflict with the previous action. Therefore, yellow change and red clearance phases are only included between sequences of conflicting actions.

Third, prior research has generated traffic using simple (Rijken, 2015; van der Pol, 2016), and likely unrealistic, models (e.g. uniform distribution vehicle generation). For reinforcement learning traffic signal control to be a contender in practice, it must be able to control intersections at any point on the demand spectrum, up to saturation demands. We train and test our agent under a dynamic rush hour demand scenario, authentically modeling the challenging environment as would be present in practice.

4. Proposed model

Attempting to solve the traffic signal control problem using reinforcement learning requires a formulation of the problem using concepts from Markov Decision Processes, specifically, defining a state space S, an action space A and a reward r.

4.1. State space

The state observation is composed of four elements. The first two elements are aggregate statistics of the intersection traffic – density and queue of each lane approaching the intersection. The density, $s_{d,l}$ is defined in Equation (4) and the queue, $s_{q,l}$, is defined in Equation (5):

$$s_{d,l} = \frac{|V_l|}{c_l} \tag{4}$$

where V_l represents the set of vehicles on lane l and c_l represents the capacity of lane l.

$$s_{q,l} = \frac{|V_{q,l}|}{c_l} \tag{5}$$

Table 1. Traffic signal phase information.

Action	NEMA phases	Compass directions	Turning movements: Left	Through	Right
NSG	2, 6	North, South	Permissive	Protected	Permissive
NSLG	1, 5	North, South	Protected	Prohibited	Permissive
EWG	4, 8	East, West	Prohibited	Protected	Permissive
EWLG	3, 7	East, West	Protected	Prohibited	Permissive

Table 2. Traffic signal phase action information.

		Selected action a_t: NSG	EWG	NSLG	EWLG
Current traffic signal phase	NSG	–	{NSY, R}	{NSLY}	{NSY, R}
	EWG	{EWY, R}	–	{EWY, R}	{EWLY}
	NSLG	–	{NSY, R}	–	{NSY, R}
	EWLG	{EWY, R}	–	{EWY, R}	–

where $V_{q,l}$ represents the set of queued (i.e. stationary) vehicles on lane l and c_l represents the capacity of lane l.

The third and fourth elements encode information about the current traffic signal phase. These elements are a one-hot vector encoding the current traffic signal phase and a real-valued scalar encoding the time spent in the current phase. A one-hot vector is a binary valued, categorical encoding of qualitative properties. Therefore, the state space at an intersection with L lanes and a set of traffic phases P is formally defined as $S \in (\mathbb{R}^L \times \mathbb{R}^L \times \mathbb{B}^{|P|} \times \mathbb{R})$. At time t, the agent observes the traffic state as $s_t \in S$.

4.2. Action space

The actions available to the agent are the green phases that the intersection traffic signal controller can enact. Although traditionally denoted numerically with natural numbers starting at 1 [i.e. National Electrical Manufacturers Association (NEMA) standard], in this research, to reduce ambiguity, we denote phases by their compass directions and movement priority. Beginning with the four compass directions, North (N), South (S), East (E), and West (W) and combining the parallel directions, along with the two different lane movements which allow vehicles to traverse the intersection, through green (G) and advanced left green (LG), the four possible actions are North–South Green (NSG), East–West Green (EWG), North–South Advance Left Green (NSLG), and East–West Advance Left Green (EWLG). For any of the action phases, all other compass direction traffic movements are prohibited (i.e. East–West Green phase imply all North–South signals are red), except for right turns, which are permissive (i.e. right on red). Formally the set of all possible actions A is defined as $A = \{\text{NSG, EWG, NSLG, EWLG}\}$, with additional information available in Table 1. Therefore, at time t, the agent chooses an action a_t, where $a_t \in A$.

However, when an agent chooses an action, it may not be immediately enacted. To ensure safe control of the intersection, additional phases may precede the chosen action. Instead of immediately transitioning from the current traffic signal phase to the selected action, a sequence of yellow change and red clearance phases may be required, dependent on the current phase and chosen action. All possible action transition sequences to transition from the current traffic phase to the chosen action phase are shown in Table 2. Note the addition of the North–South Yellow (NSY), East–West Yellow (EWY), North–South Advance Left Yellow (NSLY), East–West Advance Left Yellow (EWLY) change and red clearance (R) phases, which cannot be chosen explicitly as actions, but are part of some phase transition sequences. The set of all traffic signal phases, actions and transition phases, is defined as $P = \{\text{NSG, EWG, NSLG, EWLG, NSY, EWY, NSLY, EWLY, R}\}$.

4.3. Reward

The final element of reinforcement learning, after the agent has observed the state of the environment s, chosen an action a_t, and performed it, is receiving the reward r_t. Reward is one element that differentiates reinforcement learning from other types of machine learning; the agent seeks to develop a policy which maximizes the sum of future discounted rewards. Compared to supervised learning, in which correct actions are given by instruction, reinforcement learning has the agent evaluate actions by interacting with the environment and receiving reward.

In the context of traffic signal control, various rewards have been successfully used. Examples of rewards used are functions based on queue, delay, and throughput (El-Tantawy et al., 2014); however, there is yet no consensus on which is best.

We define the reward r_t as a function of the number of queued vehicles (6).

$$r_t = -\left(|V_q^t|^2\right) \tag{6}$$

where V_q^t is the set of queued vehicles at the intersection at time t. The number of queued vehicles is squared to increasingly penalize actions that lead to longer queues. Since reinforcement learning agents seek to maximize reward, the negative number of queued vehicles squared is used, rewarding the agent for minimizing the number of queued vehicles.

4.4. Agent

In reinforcement learning, the agent is the entity that learns by interacting with the environment. There is a single agent governing the traffic signal controller, exclusively responsible for selecting the next green traffic phase from the set of possible green phases A. We model the agent controlling the traffic signals similar to a DQN (Mnih et al., 2015). A DQN is a deep artificial neural network trained using Q-learning, a valued-based reinforcement learning algorithm. Artificial neural networks are mathematical functions inspired by biological neural networks (i.e. brains) that are appealing for their function approximation capabilities. Many problems in machine learning can suffer from the curse of dimensionality, which is when the dimensionality of the data increases, the training and computational resources required grow exponentially. Artificial neural networks have the capability to generalize from what they have learned, weakening the problems posed by the curse of dimensionality.

Artificial neural networks are defined by their architecture and weight parameters θ. Typically, neural networks are composed of at least one hidden layer of computational neurons. Additional hidden layers in a network allow it to develop high level representations of its input data through multiple layers of function composition, potentially increasing performance. The nQN-TSC is implemented as a two hidden layer artificial neural network. The input to the nQN-TSC is the state observation s_t outlined in section 4.1 State space. The two hidden layers are fully connected with 42 neurons and are each followed by rectified linear activation units (ReLu). The output layer is composed of four neurons with linear activation units, each representing the action-value of a different action (i.e. green phase).

The optimal policy π^* is achieved using the neural network to approximate the optimal action-value function by changing the parameters θ through training. The action-value function maps states to action utilities (i.e. what is the value of each action from a given state). Values represent long-term reward. If an action has a high value (and is accurately estimated), enacting it means reaping future reward. Choosing actions with the highest value will yield the optimal policy.

The specific reinforcement learning algorithm used in this research is asynchronous n-step Q-learning (Mnih et al., 2016), a variant of DQN (Mnih et al., 2015). Asynchronous n-step Q-learning differs from DQN in two fundamental ways. First, instead of a single agent interacting with a single environment, the algorithm leverages modern multi-core computer processors to simulate multiple agent-environment pairs in parallel, each agent-environment pair on a CPU thread maintaining their own local parameters θ'. The agents collect parameter updates $d\theta$, using their local parameter set θ', that are used to asynchronously update a global parameter set θ. Each agent periodically copies the global parameter set to its local parameter set every I actions. Utilizing multiple agents improves learning, as each agent is likely experiencing different environment states, taking different actions, and receiving different rewards. Second, instead of updating the parameters θ after each action, the agent executes n actions and uses data from all n actions to improve reward (and value) estimates. The asynchronous n-step Q-learning algorithm pseudo code is presented in Algorithm 1 and the model hyperparameters are presented in Table 3. Note the inclusion of reward normalization by the maximum reward $r_{\max}$ experienced thus far across all agents.

Table 3. Model hyperparameters.

Variable	Parameter	Value
$\lvert s_t \rvert$	State cardinality	42
c_l	Lane capacity	20 veh
T	Traffic simulation length	7200 s
$t_{\max}$	Max experience sequence/n-step	4
M	Total training actions	$1.5 * 10^6$
I	Update interval	7500
γ	Discount factor	0.99

The RMSProp (Tieleman & Hinton, 2012) optimization algorithm is used to train the network with an initial learning rate α of 0.001.

The agent employs an action repeat of 10 for improved stability (i.e. actions/green phases have a duration of 10 s). Yellow change and red clearance phases have a duration of 4 s. If at least one vehicle is on an incoming lane to the intersection, the agent must select an action. However, if no vehicles are present on incoming lanes, this is considered the terminal state s_{terminal}, the agent does not select an action and the traffic phase defaults to the red clearance (R) phase.

To select actions during training, we implement the simple, yet effective, ε-greedy exploration policy, which selects a random action (explore) with a probability ε and selects the action with the highest value (exploit) with a probability $1.0 - \varepsilon$. The value of ε decreases as training progresses according to (7).

$$\varepsilon_m = 1.0 - \frac{m}{M} \tag{7}$$

where m is the current number of completed actions and M is the total number of actions. Initially, $m = 0$, meaning the agent exclusively explores; however, as

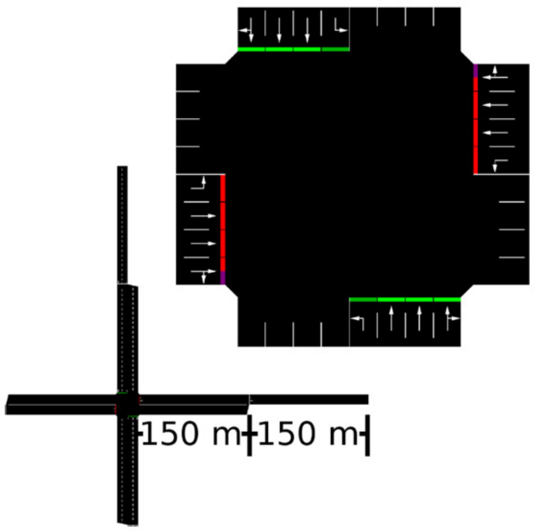

Figure 2. Simulation model of intersection, arrows represent lane movements.

training progresses, the agent increasingly exploits what it has learned, until it exclusively exploits.

5. Experimental setup and training

All experiments were conducted using the traffic microsimulator SUMO v0.29 (Krajzewicz et al., 2012). SUMO provides a Python application programing interface (API), by which custom functionality can be implemented in the traffic simulation. We used the SUMO Python API and custom code to implement all experiment code. The *n*QN-TSC was implemented using Tensorflow v1.2.1 (Abadi et al., 2015) with additional code from (Kapturowski, 2017). Additional optimized functionality was provided by NumPy and SciPy libraries (Jones, Oliphant, & Peterson, 2001). The simulations were executed using eight parallel agents on a desktop computer with an i7-6700 CPU, 16 GB of RAM running Ubuntu 16. To train the *n*QN-TSC for $M = 1.5 * 10^6$ actions requires ~6 h of wall time (Figure 1).

The intersection geometry is four lanes approaching the intersection from every compass directions (i.e. North, South, East, and West) connected to four outgoing lanes from the intersection. The traffic movements for each approach are as follows: the inner lane is left turn only, the two middle lanes are through lanes and the outer lane is through and right turning. All lanes are 300 m in length, from the vehicle origin to the intersection stop line. The state observations are derived from the stop bar to 150 m away from the intersection (i.e. only vehicles 150 m or closer to the intersection are considered in the state). A visual representation can be seen in Figure 2.

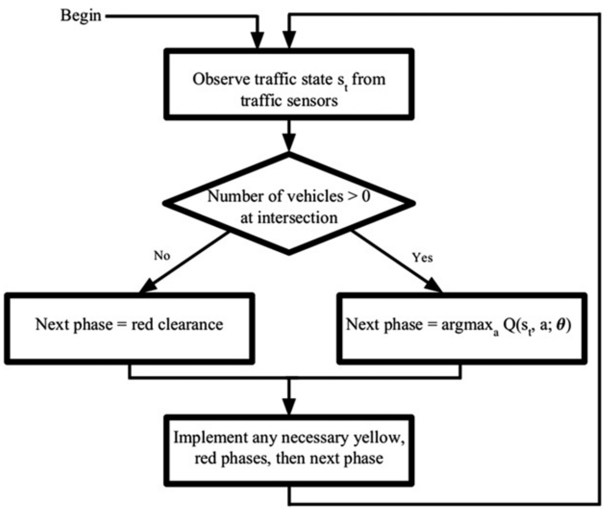
Begin
Observe traffic state s_t from traffic sensors
Number of vehicles > 0 at intersection
No
Yes
Next phase = red clearance
Next phase = argmax$_a$ Q(s_t, a; θ)
Implement any necessary yellow, red phases, then next phase

Figure 3. Block diagram describing main steps for real world deployment. The neural network $Q(s_t, a; \theta)$ is first developed in simulation and then used to determine the next phase using traffic sensor data.

For training the *n*QN-TSC, we subject it to different traffic simulations. We seek to model the dynamic nature of traffic at an intersection, which changes over time. Therefore, we model a rush hour or peak traffic demand scenario. Initially, the traffic demand is at a minimum, then, it begins increasing until it reaches a maximum, where it remains for an hour, after which it begins decreasing and then returns to a constant minimum. This training procedure is devised to create a dynamic training environment. We seek to make solving the traffic signal control problem in simulation as challenging as would be faced in practice – dynamic and stochastic. To ensure these qualities, we randomly translate the traffic demand in time at the beginning of each training simulation, displayed in Figure 4. The demands are translated to vary training and to ensure the agent does not overfit the training data. Vehicles are generated using a negative exponential distribution, where the rate parameter is used to sample from a normal distribution $\mathcal{N}(\lambda, \frac{\lambda}{10})$ to stochastically generate vehicles. This procedure is used to ensure training simulations are similar but not the same, exposing the agent to a diversity of traffic states. Training is completed after the agent has executed $M = 1.5 * 10^6$ actions. A block diagram describing the use of the proposed model after training is displayed in (Figure 3).

Vehicles are generated into the network from each compass direction with uniform probability. Vehicle routing is proportional to turning movement capacity. Vehicles are randomly assigned a route with a probability proportional to the route's lane capacity (i.e. through movements are more likely than turning movements as there are more lanes from which

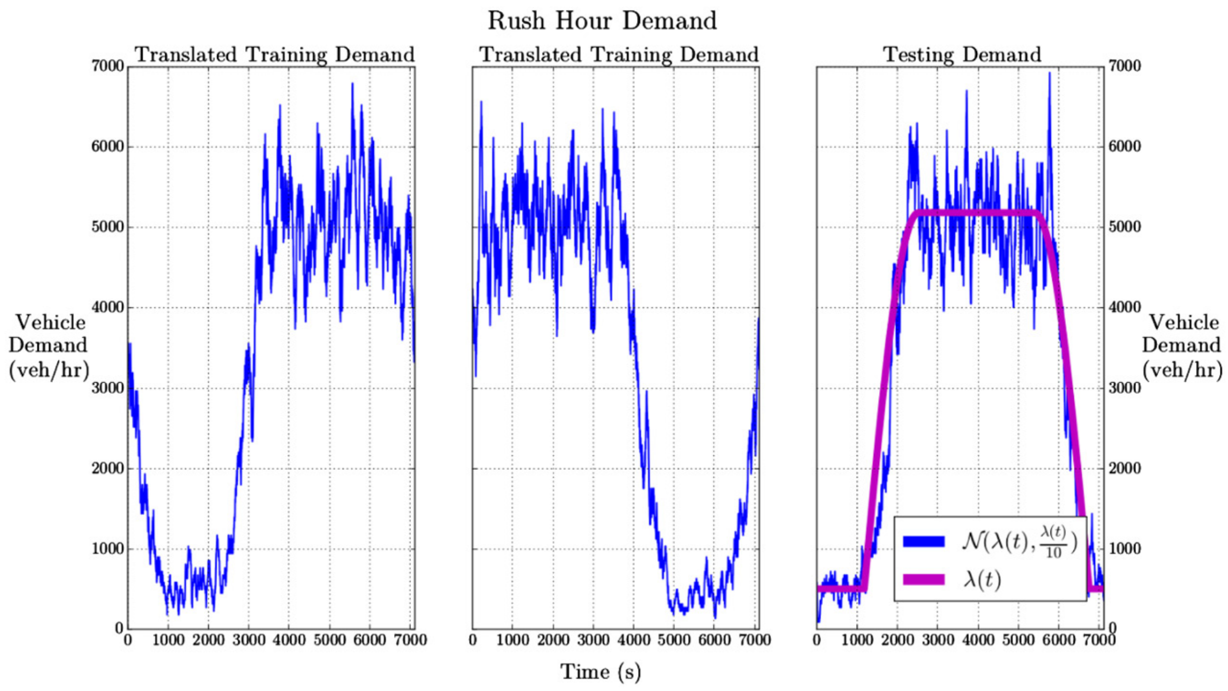

Figure 4. Examples of randomly generated simulation rush hour traffic demand. The demands are randomly translated in time for use during training while the untranslated rush hour demand is used during testing.

through movements are possible). Given the lane movements defined earlier in the section, the probability a vehicle assigned a through movement is $\frac{3}{5}$ and $\frac{1}{5}$ each for left and right turning movements.

Algorithm 1: nQN-TSC Parallel Training pseudo code

```
$m \leftarrow 0$, Initialize $\theta, \theta'$
**While** $m < M$
    $t \leftarrow 0$, Generate vehicle demands
    **While** $t < T$
        Reset parameter gradients $d\theta \leftarrow 0$
        Set thread parameters to global $\theta' = \theta$
        $t_{\text{start}} = t$
    Observe state $s_t$
    **While** $s_t \neq s_{\text{terminal}}$ **and** $t - t_{\text{start}} \neq t_{\max}$
    **If** Unif $(0, 1.0) > \varepsilon_m$
        $a_t = \text{argmax}_a Q(s_t, a; \theta')$
    **Else**
        $a_t = \text{Unif}(A)$
    Receive reward $r_t$ and observe new state $s_{t+1}$
    **If** $r_t > r_{\max}$
        $r_{\max} = r_t$
    $r_t = \frac{r_t}{r_{\max}}$
    $s_t = s_{t+1}$
    $t \leftarrow t + 1$
    $m \leftarrow m + 1$
    **If** $m \bmod I == 0$
        $\theta' \leftarrow \theta$
**If** $s_t = s_{\text{terminal}}$
    $G = 0$
**Else**
    $G = \max_a Q(s_t, a; \theta')$
**For** $i \in \{t-1, t-2, \ldots, t_{\text{start}}\}$
    $G \leftarrow r_i + \gamma G$
    Collect gradients $d\theta \leftarrow d\theta + \frac{(G - Q(s_t, a; \theta'))^2}{d\theta'}$
Asynchronously update $\theta$ with $d\theta$
```

6. Results and discussion

To evaluate the performance of the nQN-TSC after training, untranslated rush hour demands are used and its performance is compared against three other TSC methods; random, actuated and linear Q-learning.

The random TSC method enacts each action with a uniform probability (i.e. $\pi_{\text{Rand}}(s_t) = \text{Unif}(A)$). A random action policy is extremely naive with respect to optimality; however, it provides a baseline from which to compare the progress of nQN-TSC.

The actuated method is a fixed phase sequence, dynamic phase duration TSC. Sensors are used to modify the green phase durations, making this method adaptive. Each green phase has a minimum duration (10 s), after which a timer begins decrementing from a gap-out time (5 s). If a vehicle is sensed in a green phase lane (i.e. the lane currently has a permissive or protected movement given the current phase) while the timer is decrementing, the timer is reset and the current phase is temporarily extended.

Table 4. Traffic signal control rush hour results.

	(μ, σ)	
Traffic signal control method	Total throughput (veh/sim)	Total delay (s/sim)
Random	(6 508, 151)	$(81 \times 10^6,\ 20 \times 10^6)$
Actuated	(6 594, 95)	$(7.8 \times 10^6,\ 0.7 \times 10^6)$
Linear	(6 618, 91)	$(20 \times 10^6,\ 9.1 \times 10^6)$
*n*QN-TSC	(6 609, 93)	$(3.0 \times 10^6,\ 0.8 \times 10^6)$

Only a finite number of extensions can occur, limited by a maximum extension time (40 s). The sensors modeled in simulation are loop detectors (one per lane, 50 m from the stop line), which trigger when a vehicle traverses them.

The linear Q-learning method is similar to the proposed *n*QN-TSC, selecting the next phase in an *ad hoc* manner, except that it uses a linear combination of features instead of a neural network for function approximation. A linear combination of features function approximator can be thought of as a neural network without hidden layers and activation functions. Each action has its own value function, as a linear combination of features cannot approximate all action-values simultaneously. The features used by Linear Q-learning are the same as the input layer to the *n*QN-TSC, a combination of the lane queue, density, and phase information. The Linear Q-learning method does not use asynchronous learning (i.e. only one actor and one environment) or *n*-step returns, instead computing the traditional action-value target update (i.e. 1-step return). Gradient descent is used to train the Linear Q-learning action-value functions using the same model hyperparameters as the *n*QN-TSC. To train the Linear Q-learning method requires approximately 24 h of wall time.

Total vehicle delay D (8) and throughput F (9) traffic data are collected for comparing TSC methods.

$$D = \sum_{t=0}^{\mathrm{T}} \sum_{v \epsilon V^t_{\text{arrive}}} d_v \tag{8}$$

where T is the total number of simulation steps, V^t_{arrive} is the set of vehicles at time t that arrive at their destination (i.e. are removed from the simulation) and d_v is the delay experienced by vehicle v.

$$F = \sum_{t=0}^{\mathrm{T}} |V^t_{\text{arrive}}| \tag{9}$$

where V^t_{arrive} is the set of vehicles at time t that arrive at their destination (i.e. are removed from the simulation) and T is the total number of simulation steps.

To estimate relevant statistics, each TSC method is subjected to 100, 2 h simulations at testing rush hour demands. Results from the testing scenarios are shown in Table 4 and Figures 5–7.

As expected, the Random TSC achieves the worst performance, with the lowest throughput and highest delay.

Comparing the Actuated, Linear, and *n*QN-TSC, there is no significant difference in throughput; however, the *n*QN-TSC achieves the lowest delay. Interestingly, Linear Q-learning produces higher delay than the Actuated TSC. This is likely evidence that the use of a linear combination of feature as a function approximator has insufficient capacity to model the action-value function for the traffic signal control problem. Although the *n*QN-TSC and Linear Q-learning methods share many hyperparameters, the additional layers and nonlinear transformations of the neural network seem to allow it to more accurately estimate the optimal action-value function, leading to better performance.

We find it interesting that the same numbers of vehicles are traversing the intersection during the rush hour under the Actuated, Linear and *n*QN-TSC, but there are significant differences in delay. We conjecture there is a strong correlation between vehicle queues and delay, and since the *n*QN-TSC seeks to minimize the queue, it is also minimizes vehicle delay, as queued vehicles are delayed vehicles (i.e. both are stationary). The performance disparity can be attributed to the flexibility the *n*QN-TSC has in selecting the next green phase.

The *n*QN-TSC does not need to adhere to a particular phase cycle, it operates acyclically. Every action (i.e. green phase) chosen is in an attempt to maximize reward. The Actuated TSC does not exhibit such goal-oriented behavior, instead executing primitive logic that produces inferior performance compared to the *n*QN-TSC. Granted, the Actuated TSC does not have the benefit of training like the *n*QN-TSC, but its design precludes it from being of any benefit even if it was available. The information contained in the *n*QN-TSC's state representation is also much higher compared to the Actuated TSC, improving its action selection. The Actuated TSC's state representation is a bit for every lane and this low-resolution state likely aliases many different traffic states. These results provide evidence to support that adaptive traffic signal control can be accomplished with reinforcement learning and function approximation.

We also present individual vehicle delay results for each lane/turning movement in Figure 7. We observe that although in aggregate the *n*QN-TSC achieves the lowest total delay results, this observation does not exist for all lanes. For the through and right lanes, the *n*QN-TSC exhibits the lowest median delay, quartiles

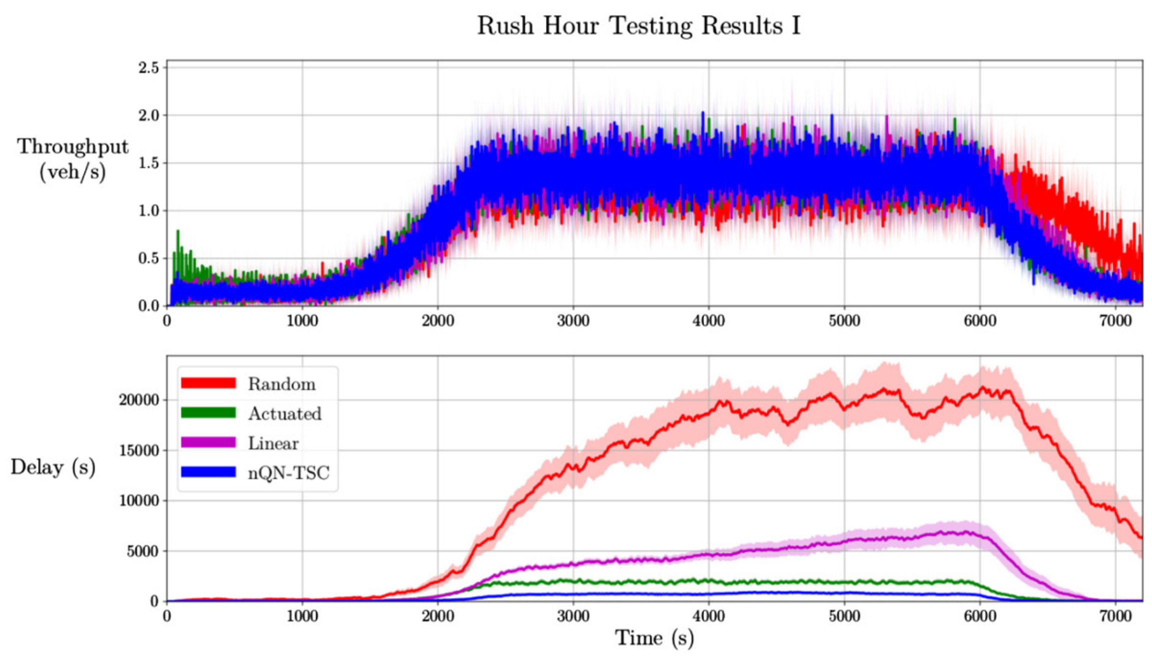

Figure 5. Performance of TSC methods under 100, 2 h rush hour simulations. Lines represent the mean and the shaded area is the 99% confidence interval. Note that the throughput is influenced by the traffic demand. Traffic demand begins low, increases to a maximum, remains constant, and then decreases at the end of the testing scenario.

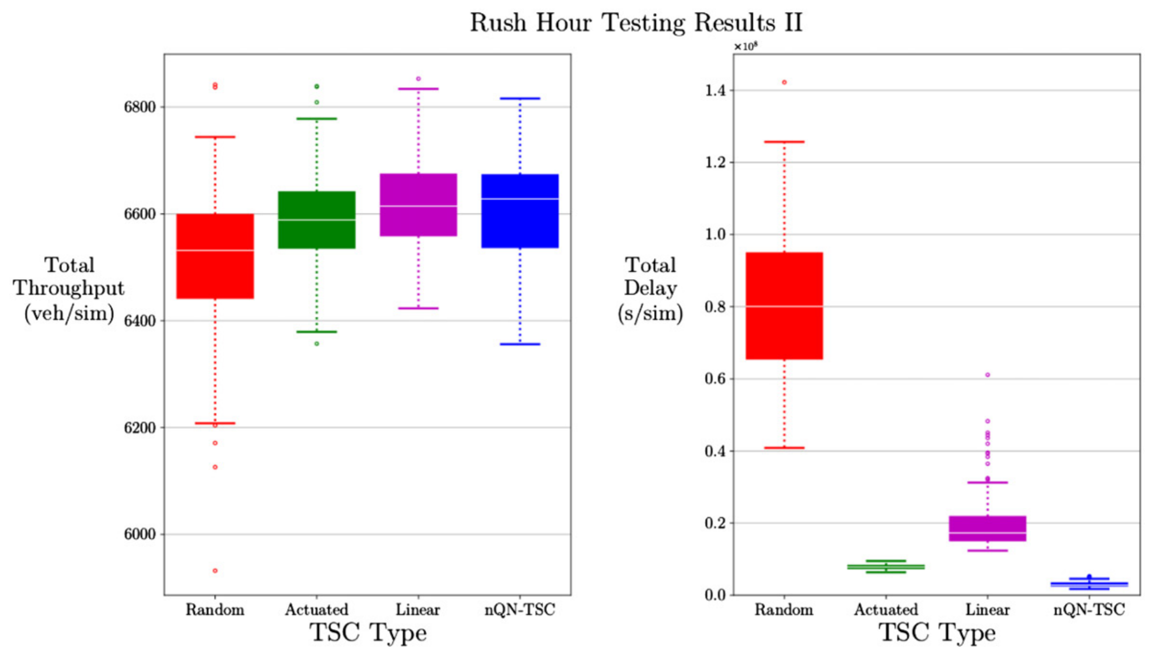

Figure 6. Total throughput and delay for individual simulation runs. Colored rectangles represent the range between the first and third quartiles, solid white line inside the rectangle represents the median, dashed colored lines represent 1.5 × interquartile range, colored dots represent outliers.

and outliers compared to the other TSC methods. However, observing the left lane delay, the Actuated method achieves lower delay than the *n*QN-TSC, specifically in regards to outliers, as the *n*QN-TSC has significant outliers which indicate some left turning vehicles have to wait a disproportionate amount of time to turn left compared to other vehicles. This result can be understood by considering the reward function, which seeks to minimize the squared number of queued vehicles. The intersection geometry used in this research has, in any single compass direction, three incoming lanes which allow through movements while only one lane for left turn movements. The *n*QN-TSC has discovered that selecting

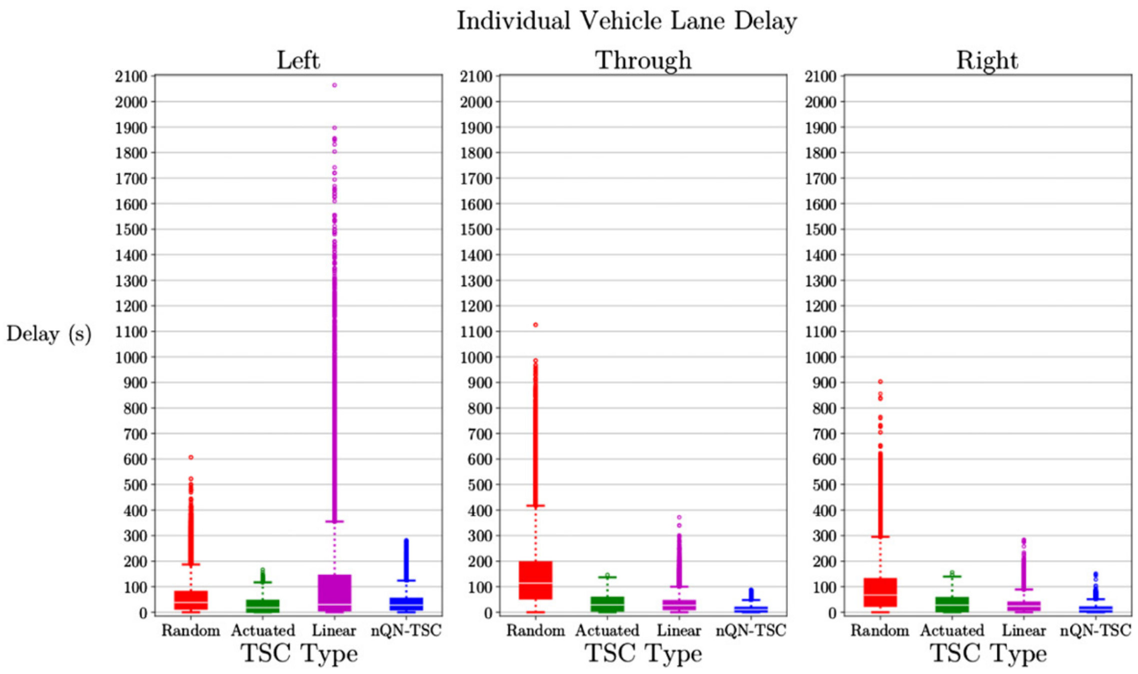

Figure 7. Individual vehicle delay by intersection lane movement and TSC type. Colored rectangles represent the range between the first and third quartiles, solid white line inside the rectangle represents the median, dashed colored lines represent 1.5 × interquartile range, colored dots represent outliers. Performance of TSC methods under 100, 2 h rush hour simulations.

the actions/phases which give protected green signals to through movements (i.e. NSG and EWG) yield the highest reward. The *n*QN-TSC learns to favor these actions over the exclusive left turning actions/phases (i.e. NSLG, EWLG), because they yield less reward from having fewer queued vehicles potentially traversing the intersection when selected.

The *n*QN-TSC's behavior is contrasted with the Actuated method, which implements a cycle and reliably enacts exclusive left turning phases, yielding lower delays for vehicles in left turning lanes. This is an example of a reinforcement learning agent developing a policy which seems optimal with respect to the reward, but with undesirable, unintended consequences. This behavior could be corrected with a different reward function for the *n*QN-TSC, but perhaps at the expense of its desirable performance in the other metrics (e.g. total delay, through lane delay, right lane delay). This is an interesting result that we have not observed discussed in the TSC reinforcement learning literature and should be the focus of future research.

7. Conclusions and future work

We modeled an *n*-step Q-network traffic signal controller (*n*QN-TSC) trained to control phases at a traffic intersection to minimize vehicle queues. Compared to a loop detector Actuated TSC in a stochastic rush hour demand scenario, the modeled *n*QN-TSC achieve superior performance by reducing average total vehicle delay by ∼40%. This research provides evidence that valued-based reinforcement learning methods with function approximation can provide improved performance compared to established traffic signal control methods in stochastic environments, such as rush hour demands.

Beyond this research, many areas of future work are available. The function approximator used in this research was simple (i.e. two fully connected hidden layer neural network) compared to research in other domains; using convolutional or recurrent layers in the neural network will likely yield additional performance improvements. Future work can also investigate policy or actor-critic reinforcement learning algorithms, different state representations, pedestrians, and multiple intersections. These ideas will be explored in future endeavors by the authors.

Acknowledgments

The authors would like to acknowledge the authors of the various programing libraries used in this research. The authors would also like to acknowledge the comments and support of friends, family, and colleagues.

ORCID

Wade Genders http://orcid.org/0000-0001-9844-8652

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., … Zheng, X. (2017). Tensorflow: Large-scale machine learning on heterogeneous systems (1.2.1). Retrived from https://tensorflow.org.

Abdoos, M., Mozayani, N., & Bazzan, A. L. (2013). Holonic multi-agent system for traffic signals control. *Engineering Applications of Artificial Intelligence*, *26*(5–6), 1575–1587. doi: 10.1016/j.engappai.2013.01.007

Abdulhai, B., Pringle, R., & Karakoulas, G. J. (2003). Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering*, *129*(3), 278–285. doi: 10.1061/(ASCE)0733-947X(2003)129:3(278)

Arel, I., Liu, C., Urbanik, T., & Kohls, A. (2010). Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems*, *4*(2), 128–135. doi: 10.1049/iet-its.2009.0070

Aziz, H. A., Zhu, F., & Ukkusuri, S. V. (2018). Learning based traffic signal control algorithms with neighborhood information sharing: An application for sustainable mobility. *Journal of Intelligent Transportation Systems*, *22*(1), 40–52. doi: 10.1080/15472450.2017.1387546

Balaji, P., German, X., & Srinivasan, D. (2010). Urban traffic signal control using reinforcement learning agents. *IET Intelligent Transport Systems*, *4*(3), 177–188. doi: 10.1049/iet-its.2009.0096

Brockfeld, E., Barlovic, R., Schadschneider, A., & Schreckenberg, M. (2001). Optimizing traffic lights in a cellular automaton model for city traffic. *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics*, *64*(5 Pt 2), 056132.

Casas, N. (2017). Deep deterministic policy gradient for urban traffic light control. arXiv preprint arXiv:1703.09035.

Chin, Y. K., Bolong, N., Kiring, A., Yang, S. S., & Teo, K. T. K. (2011). Q-learning based traffic optimization in management of signal timing plan. *International Journal of Simulation, Systems, Science & Technology*, *12*(3), 29–35.

El-Tantawy, S., Abdulhai, B., & Abdelgawad, H. (2013). Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): Methodology and large-scale application on downtown toronto. *IEEE Transactions on Intelligent Transportation Systems*, *14*(3), 1140–1150. doi: 10.1109/TITS.2013.2255286

El-Tantawy, S., Abdulhai, B., & Abdelgawad, H. (2014). Design of reinforcement learning parameters for seamless application of adaptive traffic signal control. *Journal of Intelligent Transportation Systems*, *18*(3), 227–245. doi: 10.1080/15472450.2013.810991

Gregoire, J., Qian, X., Frazzoli, E., De La Fortelle, A., & Wongpiromsarn, T. (2015). Capacity-aware backpressure traffic signal control. *IEEE Transactions on Control of Network Systems*, *2*(2), 164–173. doi: 10.1109/TCNS.2014.2378871

Hunt, P., Robertson, D., Bretherton, R., & Winton, R. (1981). *Scoot-a traffic responsive method of coordinating signals. (Report No. LR 1014)*. Berkshire, UK: Transport and Road Research Laboratory.

Jones, E., Oliphant, T., & Peterson, P., (2001). *SciPy: Open source scientific tools for Python (0.19.0)*. Retrieved from https://scipy.org.

Kapturowski, S. (2017). *Tensorflow-rl*. Retrieved from https://github.com/steveKapturowski/tensorflow-rl

Khamis, M. A., & Gomaa, W. (2012). Enhanced multiagent multi-objective reinforcement learning for urban traffic light control. Paper presented at the 2012 11th International Conference on Machine Learning and Applications (ICMLA), IEEE, Vol. *1*, pp. 586–591.

Khamis, M. A., & Gomaa, W. (2014). Adaptive multiobjective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework. *Engineering Applications of Artificial Intelligence*, *29*, 134–151. doi: 10.1016/j.engappai.2014.01.007

Khamis, M. A., Gomaa, W., El-Mahdy, A., & Shoukry, A. (2012). Adaptive traffic control system based on bayesian probability interpretation. Paper presented at the 2012 Japan-Egypt Conference on Electronics, Communications and Computers (pp. 151–156). Alexandria, Egypt: IEEE.

Khamis, M. A., Gomaa, W., & El-Shishiny, H. (2012). Multi-objective traffic light control system based on bayesian probability interpretation. Paper presented at the 2012 15th International IEEE Conference on, Intelligent Transportation Systems (ITSC), IEEE, pp. 995–1000.

Krajzewicz, D., Erdmann, J., Behrisch, M., & Bieker, L. (2012). Recent development and applications of SUMO—Simulation of Urban MObility. *International Journal on Advances in Systems and Measurements*, *5*(3 & 4), 128–138.

Lee, J., Abdulhai, B., Shalaby, A., & Chung, E.-H. (2005). Real-time optimization for adaptive traffic signal control using genetic algorithms. *Journal of Intelligent Transportation Systems*, *9*(3), 111–122. doi: 10.1080/15472450500183649

Li, L., Lv, Y., & Wang, F.-Y. (2016). Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, *3*(3), 247–254. doi: 10.1109/JAS.2016.7508798

Lowrie, P. (1990). Scats, sydney co-ordinated adaptive traffic system: A traffic responsive method of controlling urban traffic. (Report No. N/A). New South Wales, Australia: Traffic Control Section Roads and Traffic Authority of New South Wales.

Mannion, P., Duggan, J., & Howley, E. (2016). An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In *Autonomic road transport support systems* (pp. 47–66). Basel, Switzerland: Springer.

Mirchandani, P., & Head, L. (2001). A real-time traffic signal control system: Architecture, algorithms, and analysis. *Transportation Research Part C: Emerging Technologies*, *9*(6), 415–432. doi: 10.1016/S0968-090X(00)00047-4

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., … Hassabis, D. (2016). Asynchronous methods for deep reinforcement learning. Paper presented at the International Conference on Machine Learning, pp. 1928–1937. New York, NY: International Machine Learning Society..

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., … Ostrovski, G. (2015). Human level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. doi: 10.1038/nature14236

Mousavi, S. S., Schukat, M., Corcoran, P., & Howley, E. (2017). *Traffic light control using deep policy-gradient and value-function based reinforcement learning*. Retrieved from https://arxiv.org/abs/1704.08883

Prashanth, L., & Bhatnagar, S. (2011). Reinforcement learning with function approximation for traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, *12*(2), 412–421. doi: 10.1109/TITS.2010.2091408

Rijken, T. (2015). *Deeplight: Deep reinforcement learning for signalised traffic control* (Master's thesis). London, UK: University College London.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*(1), 9–44.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction* (Vol. *1)*. Cambridge: MIT Press.

Thorpe, T. L., & Anderson, C. W. (1996). *Traffic light control using sarsa with three state representations*. Technical Report, Citeseer.

Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, *4*(2), 26?31.

Timotheou, S., Panayiotou, C. G., & Polycarpou, M. M. (2015). Distributed traffic signal control using the cell transmission model via the alternating direction method of multipliers. *IEEE Transactions on Intelligent Transportation Systems*, *16*(2), 919–933.

van der Pol, E. (2016). *Deep reinforcement learning for coordination in traffic light control* (Master's thesis). Amsterdam, Netherlands: University of Amsterdam.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*(3–4), 279–292. doi: 10.1007/BF00992698

Wiering, M., et al. (2000). *Multi-agent reinforcement learning for traffic light control. Proceedings of the 17th International Conference on Machine Learning. (pp. 1151–1158). San Francisco, CA: International Machine Learning Society*, pp. 1151–1158.

Wongpiromsarn, T., Uthaicharoenpong, T., Wang, Y., Frazzoli, E., & Wang, D. (2012). Distributed traffic signal control for maximum network throughput. Paper presented at the 2012 15th International IEEE Conference on Intelligent Transportation Systems (ITSC), IEEE, pp. 588–595.