Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 130 (2018) 26–33

Procedia Computer Science

www.elsevier.com/locate/procedia

The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018)

Evaluating reinforcement learning state representations for adaptive traffic signal control

Wade Genders[a], Saiedeh Razavi[a]

[a]*Civil Engineering, McMaster University, 1280 Main St. West, Hamilton, L8S 4L8, Canada*

Abstract

Reinforcement learning has shown potential for developing effective adaptive traffic signal controllers to reduce traffic congestion and improve mobility. Despite many successful research studies, few of these ideas have been implemented in practice. There remains uncertainty about what the requirements are in terms of data and sensors to actualize reinforcement learning traffic signal control. We seek to understand the data requirements and the performance differences in different state representations for reinforcement learning traffic signal control. We model three state representations, from low to high-resolution, and compare their performance using the asynchronous advantage actor-critic algorithm with neural network function approximation in simulation. Results show that low-resolution state representations (e.g., occupancy and average speed) perform almost identically to high-resolution state representations (e.g., individual vehicle position and speed). These results indicate implementing reinforcement learning traffic signal controllers may be possible with conventional sensors, such as loop detectors, and do not require sophisticated sensors, such as cameras or radar.

Peer-review under responsibility of the Conference Program Chairs.

Keywords: Adaptive Traffic Signal Control; Reinforcement Learning; Artificial Neural Networks; Intelligent Transportation Systems; Traffic Simulation;

1. Introduction

Vehicle congestion is a major problem in cities across the world. Developing additional infrastructure is expensive and a protracted process which can exacerbate the problem until completed. Instead of adding more infrastructure, another solution is to optimize currently available infrastructure. Intersection traffic signal controllers (TSC) are ubiquitous in modern road infrastructure and their functionality greatly impacts all users. Many research studies have proposed improvements to TSC, broadly in an attempt to make them adaptive to current traffic conditions. Reinforcement learning has been shown to be effective in developing adaptive TSC with many research studies detailing

* Corresponding author: Wade Genders
E-mail address: genderwt@mcmaster.ca

1877-0509
Peer-review under responsibility of the Conference Program Chairs.
10.1016/j.procs.2018.04.008

promising results. Despite the encouraging research, few reinforcement learning adaptive TSC have been deployed in the field. One inhibiting factor is the resources required; to observe the traffic state, reinforcement learning TSC often require high-resolution data beyond the detection capability of traditional sensors (i.e., loop detectors). This research focuses on the potential state definitions of reinforcement learning TSC and ascertaining the performance differences between them. We seek to answer, can a reinforcement learning TSC function using low-resolution data from traditional sensors such loop detectors? Or is high-resolution data from sophisticated sensors (e.g., cameras, radar) required? Answering this question will help individuals interested in deploying reinforcement learning TSC in the field, as they will be aware of the requirements and potential outcomes. We use the traffic microsimulator SUMO[1] and the asynchronous advantage actor-critic (A3C) algorithm[2] to train and evaluate multiple adaptive TSC with different resolution state representations.

2. Literature Review

Many research studies have recognized and displayed reinforcement learning's capability for providing a solution to TSC. Early research provided proof-of-concept for reinforcement learning in TSC[3,4,5,6]. Later research applied reinforcement learning methods to more realistic and complex traffic models[7,8,9,10,11,12]. Developments in machine learning have yielded deep reinforcement learning techniques[2,13,14] which have subsequently been applied for TSC[15,16,17,18,19].

Considering the aforementioned research and the extensive reinforcement learning TSC reviews[20,21,22], we identify numerous possible state representations: vehicle density, flow, queue, location, speed along with the current traffic phase, cycle length and red time. These state representations form a resolution spectrum of the current traffic state, from coarse (e.g., flow) to fine (e.g., individual vehicle position and speed). We consider state representation across the resolution spectrum, requiring different sensors, and compare their performance. The results can guide individuals interested in practical implementation.

3. Model

3.1. Reinforcement Learning

Reinforcement learning is a type of machine learning for solving sequential decision-making problems[23]. A reinforcement learning agent learns a policy $\pi(s) = a$, mapping from states s to actions a, to achieve a goal in an environment under uncertainty. Through repeated environment interactions, a reinforcement learning agent strives to develop an optimal policy π^*, which maximizes the sum of future discounted ($\gamma \in (0, 1]$) rewards, defined as the return G_t in Equation 1:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{1}$$

The agent interacts with the environment in repeating sequences of, at time t, observing the environment state s_t, taking action a_t, receiving reward r_t and entering a new state s_{t+1}. Over time, the agent learns what actions in what states maximize long-term reward, also known as value. Rewards quantitatively represent how successful the agent's policy is achieving its mandated goal.

The A3C algorithm is used to develop parameterized θ policy $\pi(a|s;\theta)$ (Equation 2) and value $V^\pi(s;\theta)$ functions (Equation 3). The agent develops a value function (critic), which estimates the expected return from a given state, which is used to improve the policy (actor).

$$\pi(a|s;\theta) = \Pr[a_t = a|s_t = s;\theta] \tag{2}$$

$$V^\pi(s;\theta) = \mathbb{E}[G_t|s_t = s;\theta] \tag{3}$$

The parameters are used for neural network function approximation, defining the weights between neurons.

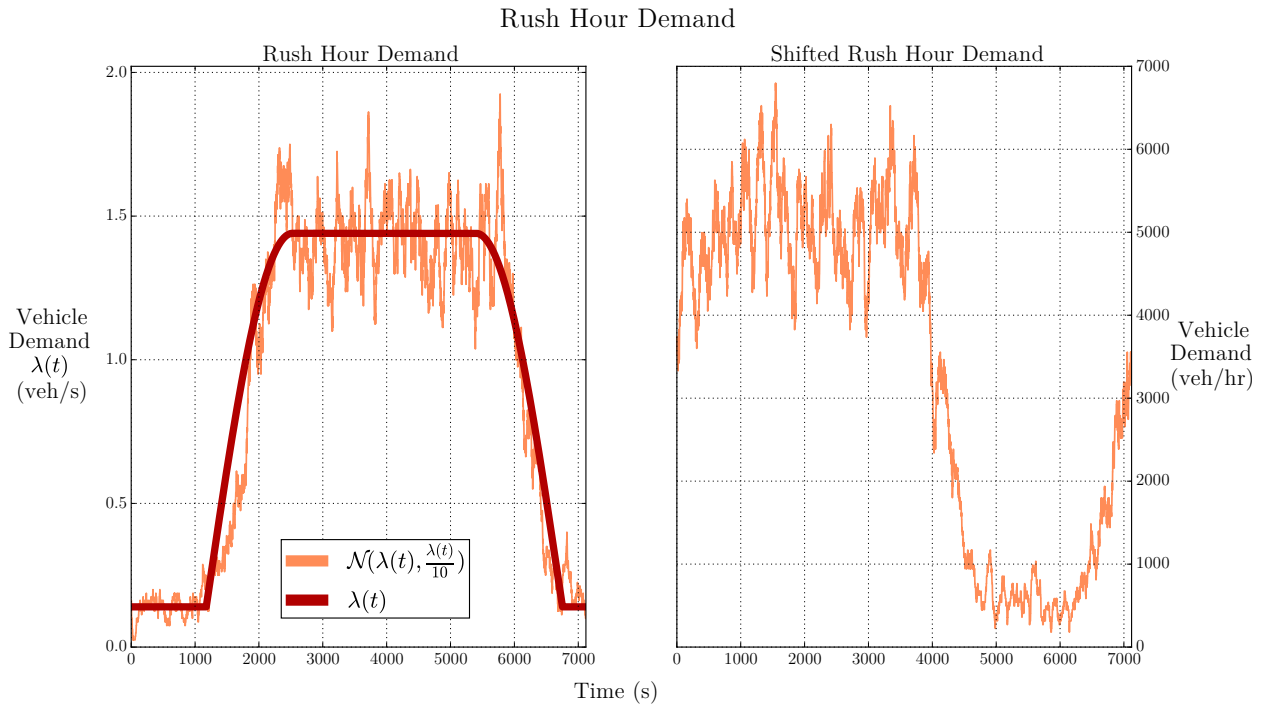

Fig. 1. Rush hour demand traffic scenario used for training (right) and testing (left).

3.2. Environment

The environment used to train the reinforcement learning adaptive TSC is the traffic microsimulator SUMO [1]. The network geometry is an isolated intersection with four origin-destination zones in each compass direction; North (N), South (S), East (E) and West (W). Each origin-destination zone is connected to the intersection with eight lanes, four incoming and four outgoing. The turning movements for incoming lanes to the intersection are; the right lane allows right turn and through movements, the middle two lanes allow through movements and the left lane allows left turns.

We simulate a peak or rush hour traffic demand scenario for training. The traffic is generated stochastically using a negative exponential distribution with a rate parameter λ. To add further stochasticity, the rate parameter λ is sampled from a normal distribution $\mathcal{N}(\lambda, \frac{\lambda}{10})$, as seen in Figure 1.

3.3. State

At time t the reinforcement learning agent observes the state of the environment s_t. The agent's behaviour and ability to learn is greatly influenced by the state and its definition. Three state spaces are defined for reinforcement learning TSC with different resolutions of the environment. All state representations include the most recent traffic phase encoded as a one-hot vector (i.e. a phase from the set of all traffic phases P) and the time spent in that phase. One-hot vectors are used to represent categorical variables. For K categories, a $K \times K$ identity matrix represents all the categories, with each row representing a different category.

3.3.1. Occupancy and Speed

The lowest resolution state space is defined using occupancy and average speed. Loop detectors are the most common sensors at intersections and can be used to collect coarse traffic statistics, such as occupancy and average speed for each lane. We model each incoming lane with two loop detectors, one at the stop line and the other setback 50 m from the stop line. The occupancy and average speed (normalized by the speed limit) are computed from the the previous 10 s interval. Given m incoming lanes, the loop-TSC state is $s_t \in \mathbb{R}^{(2m+|P|+1)}$.

3.3.2. Queue and Density

A higher resolution state space is defined using vehicle density and queue. Loop detectors are inadequate to collect queue and density data reliably, more sophisticated sensors are required (i.e., video cameras, radar). Assuming a jam density k_j, V_l represents the set of vehicles on lane l and $V_{l,q}$ the set of queued vehicles on lane l, we define vehicle lane density $\frac{V_l}{k_j}$ and vehicle lane queue $\frac{V_{l,q}}{k_j}$. Given m incoming lanes, we denote the queue-TSC state $s_t \in \mathbb{R}^{(2m+|P|+1)}$.

3.3.3. Discrete Cell Encoding

The highest resolution state space discretizes each incoming lane into cells of a fixed length $c = 2.5$ m, termed the discrete traffic state encoding (dtse). Cells are binary encoded, 1 represents the presence of a vehicle and 0 represents the absence of a vehicle. Sophisticated sensors (i.e., video cameras, radar) would also be required to collect this state data. Given an incoming lane length $L = 135$ m, the dtse-TSC state is $s_t \in \mathbb{R}^{(\frac{L}{c}m+|P|+1)t}$, where we use a history of the $t = 2$ most recent states. This state definition was first proposed using SARSA reinforcement learning [3] and has since been utilized in deep reinforcement learning TSC [15,16,18].

3.4. Actions

After observing state s_t, the agent chooses an action $a_t \in A$. The actions available are the green traffic phases, denoted in this research by a pair of compass directions and set of movement priorities (i.e., G represents protected through movements and permissive left turn movements and LG represents protected left turn movements and prohibited through movements). For example, the action NSG represents North-South protected through movements, permissive left turn movements and prohibits all East-West movements. Actions in a sequence may require yellow change and red clearance phases, with additional action/phase information detailed in Table 1. The set of all possible actions is denoted $A = \{\text{NSG, EWG, NSLG, EWLG}\}$. All actions $a_t \in A$ have a duration of 10 s and yellow and red phases have a duration of 4 s. When no vehicles are present at the intersection (i.e., $\forall l, V_l =)$), all movements are prohibited with the red clearance phase.

The agent's traffic signal control policy is acyclic, unconstrained and *ad-hoc*. We argue imposing a cycle in reinforcement learning TSC is presumptuous. If a cycle is optimal, the agent will develop such a policy. There are no maximum times for each phase and the agent chooses the next action/phase without limitation.

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	Permissive	Protected	Permissive
EWLG	3, 7	East, West	Protected	Prohibited	Permissive

3.5. Reward

After observing state s_t and taking action a_t, the agent receives a scalar reward $r_t \in \mathbb{R}$ from the environment. The reward is feedback for how 'good' action a_t was in s_t. Many rewards have been proposed for reinforcement learning TSC (e.g., functions of throughput, queue, delay). We define reward in Equation 4 as change in cumulative delay:

$$r_t = D_{a_t} - D_{a_{t+1}} \tag{4}$$

Where $D_{a_t}, D_{a_{t+1}}$ represent the cumulative delay at the intersection when action a_t and a_{t+1} are taken. Cumulative vehicle delay D at time t is defined in Equation 5:

$$D_t = \sum_{v \in V_t} d_t^v \tag{5}$$

Where V_t is the set of vehicles on incoming lanes in the simulation at time t and d_t^v is the delay of vehicle v at time t.

3.6. Agent

The agent is the entity, through repeated interaction with the environment, that implements and improves the policy π. In this research, the agent chooses the next green traffic phase. We model the agent as an artificial neural network and train it using the A3C algorithm.

An artificial neural network is chosen for its flexible function approximation capabilities. The architecture of the neural network is an input layer and then a fully connected hidden layer with rectified linear (ReLu) activation functions followed by another fully connected hidden layer with ReLu activation functions. The output layer has $|A| = 4$ neurons with softmax activation functions which output the action probabilities representing the policy. The number of neurons in each hidden layer for each state representation is equal to the cardinality of the input state; loop-TSC and queue-TSC hidden layers have 42 neurons and dtse-TSC hidden layers have 1780 neurons.

The A3C algorithm simulates multiple actor-critic agents in parallel, each with their own environment. Using a local parameter set θ', each agent computes an advantage $Adv = G_t - V(s;\theta')$ from experience sequences of length $t_{max} = 32$. The advantage is a measure of the difference between the actual and expected performance of the policy. The advantage is used to compute parameter gradients $d\theta$ which are asynchronously applied to a global parameter set θ. Each agent periodically copies the global parameters θ as their local parameters θ'. The rewards are standardized before computing the return (i.e., $r_t \leftarrow \frac{r_t - \mu_r}{\sigma_r}$) and the gradient of the policy entropy $\beta\nabla_{\theta'} H(\pi(s_i;\theta'))$ is added to the parameter update for improved learning.

Algorithm 1: Asynchronous Advantage Actor-Critic training pseudocode[2]

$n \leftarrow 0$, Initialize $\theta, \theta^{'}$
While $n < N$
  $t \leftarrow 0$, Generate sim with randomly shifted demand
  While $t < T$
    Reset parameter gradients $\mathrm{d}\theta \leftarrow 0$
    Set thread parameters to global $\theta' = \theta$
    $t_{start} = t$
    Observe state s_t
    While $s_t \neq s_{terminal}$ **and** $t - t_{start} \neq t_{max}$
      Perform action a_t from $\pi(a_t|s_t;\theta')$
      Receive reward r_t and observe new state s_{t+1}
      $s_t = s_{t+1}$
      $t \leftarrow t + 1$
    If $s_t == s_{terminal}$
      $G_t = 0$
    Else
      $G_t = V(s_t;\theta')$
    For $i \in \{t-1, t-2, ..., t_{start}\}$
      $G_t \leftarrow r_i + \gamma G$
      $Adv = G_t - V(s_i;\theta')$
      $\epsilon = \beta\nabla_{\theta'} H(\pi(s_i;\theta'))$
      Collect policy gradients $\mathrm{d}\theta \leftarrow \mathrm{d}\theta + \nabla_{\theta'} \log_e \pi(a_i|s_i;\theta')(Adv) + \epsilon$
      Collect value gradients $\mathrm{d}\theta \leftarrow \mathrm{d}\theta + \frac{\partial (Adv)^2}{\partial \theta'}$
    Asynchronously update θ with $\mathrm{d}\theta$
  $n \leftarrow n + 1$

4. Experiments

We subject each different state representation reinforcement learning TSC to $N = 1\,000$ rush hour demand scenarios, $T = 7\,200$ simulation steps in duration, for training. As a form of data augmentation and to prevent overfitting, the rush hour demand scenario is randomly shifted in time for each simulation, displayed in Figure 1. Training is conducted using a consumer i7 CPU with 8 parallel, asynchronous actor-critics, each with their own environment. Training for each state representation was 3-6 hours wall clock time.

For comparison after training, each different state reinforcement learning TSC is subjected to 100 rush hour demand scenarios without data augmentation to ensure parity during testing. Throughput, delay and queue statistics are collected during testing simulations.

The traffic microsimulator SUMO[1] is used for all simulations. Tensorflow[24], SciPy[25] and additional Python libraries[26] are used for implementing the neural networks and reinforcement learning.

As a baseline for comparison, we model an Actuated TSC which uses loop detectors to modulate the green phase lengths. The Actuated TSC is cyclic; each phase has a minimum green time of 10 s, after which a gap-out timer begins decrementing from 5 s. If a vehicle is detected in a lane with a protected movement under the current phase the gap-out timer is reset to 5 s, up to a maximum of 40 s.

5. Results

Testing results are displayed in Table 2 and visually in Figure 2. All reinforcement learning TSC achieve superior performance in reducing delay and queue lengths compared to the Actuated TSC. This result is not surprising, as the reinforcement learning TSC have greater flexibility in action selection compared to the Actuated TSC, which has to implement phases in a cycle.

Observing the traffic metrics collected, there appears to be little to no difference between the different state representations. There is no difference in throughput or queue however the loop-TSC exhibits the highest delay compared to the queue-TSC and dtse-TSC. This result is surprising to the authors, specifically how little difference there is between the different state representations considering the spectrum of data resolution modeled. Results from research in other domains using deep reinforcement learning suggest that high-resolution state data can yield superior performance to low-resolution state data. In this research, we find very little difference in performance despite the disparity in data resolution between the different state representations.

Table 2. Cumulative statistics testing results comparing state definitions.

Traffic Signal Control Method	Total Throughput (veh/sim)	$(\hat{\mu}, \hat{\sigma})$ Total Delay (s/sim)	Total Queue (veh/sim)
loop-TSC	$(6\,609, 86)$	$(2.8 \times 10^6, 4.6 \times 10^5)$	$(1.2 \times 10^5, 0.8 \times 10^4)$
queue-TSC	$(6\,612, 87)$	$(2.3 \times 10^6, 2.5 \times 10^5)$	$(1.2 \times 10^5, 0.7 \times 10^4)$
dtse-TSC	$(6\,598, 92)$	$(2.4 \times 10^6, 4.9 \times 10^6)$	$(1.1 \times 10^5, 0.9 \times 10^4)$
Actuated	$(6\,529, 98)$	$(7.7 \times 10^6, 7.3 \times 10^5)$	$(2.2 \times 10^5, 1.2 \times 10^4)$

6. Conclusion

We modeled adaptive TSC using the A3C reinforcement learning algorithm with various state representations to determine any differences in performance. Results show that data collected from traditional and ubiquitous sensors such as loop detectors are sufficient for reinforcement learning adaptive TSC. Under the model devised in this research, high-resolution state representations requiring sophisticated sensors offer improvements only by reducing delay by approximately 10%, with no difference in queue or throughput.

A potential explanation for the lack of performance disparity is that high-resolution state data may only yield benefits with sufficiently complex function approximators. The fully connected neural network architecture may be too simple compared to convolutional, long short-term memory layers or residual connections. These architectures

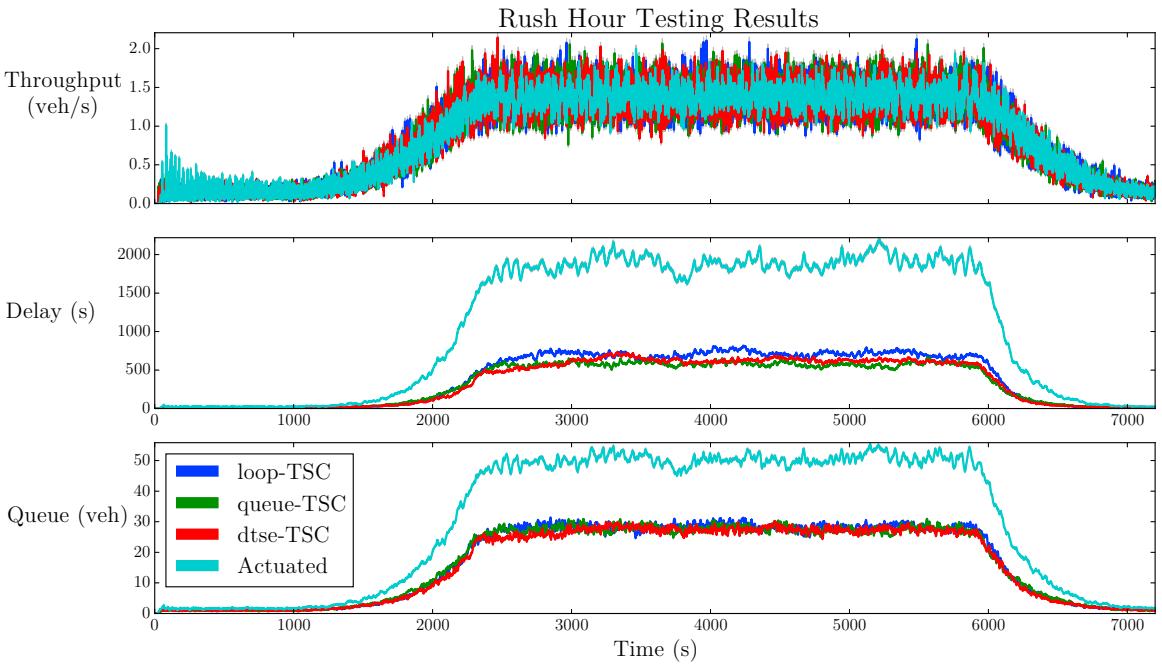

Fig. 2. Testing results comparing state definitions.

may be required to leverage high-resolution state data and develop the rich representations necessary for improved performance. Given the lack of substantial performance between state representations, this may be evidence that sophisticated machine learning methods, such as reinforcement learning, are unnecessary for traffic signal control. Other, less resource intensive machine learning methods such as decision trees or regression methods may be more appropriate. These methods may offer similar performance while also being much more comprehensible, as opposed to the black box nature of a neural network. These ideas should be should be considered in future research.

References

1. Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L.. Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements* 2012;**5**(3&4):128–138.
2. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., et al. Asynchronous methods for deep reinforcement learning. In: *International Conference on Machine Learning*. 2016, p. 1928–1937.
3. Thorpe, T.L., Anderson, C.W.. Traffic light control using sarsa with three state representations. Tech. Rep.; Citeseer; 1996.
4. Wiering, M., et al. Multi-agent reinforcement learning for traffic light control. In: *ICML*. 2000, p. 1151–1158.
5. Brockfeld, E., Barlovic, R., Schadschneider, A., Schreckenberg, M.. Optimizing traffic lights in a cellular automaton model for city traffic. *Physical Review E* 2001;**64**(5):056132.
6. Abdulhai, B., Pringle, R., Karakoulas, G.J.. Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering* 2003;**129**(3):278–285.
7. Wiering, M., Vreeken, J., Van Veenen, J., Koopman, A.. Simulation and optimization of traffic in a city. In: *Intelligent Vehicles Symposium, 2004 IEEE*. IEEE; 2004, p. 453–458.
8. El-Tantawy, S., Abdulhai, B., Abdelgawad, H.. 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* 2013;**14**(3):1140–1150.
9. Abdoos, M., Mozayani, N., Bazzan, A.L.. Holonic multi-agent system for traffic signals control. *Engineering Applications of Artificial Intelligence* 2013;**26**(5):1575–1587.
10. Zhu, F., Aziz, H.A., Qian, X., Ukkusuri, S.V.. A junction-tree based learning algorithm to optimize network wide traffic control: A coordinated multi-agent framework. *Transportation Research Part C: Emerging Technologies* 2015;**58**:487–501.
11. Jin, J., Ma, X.. A group-based traffic signal control with adaptive learning ability. *Engineering applications of artificial intelligence* 2017;**65**:282–293.
12. Aslani, M., Mesgari, M.S., Wiering, M.. Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events. *Transportation Research Part C: Emerging Technologies* 2017;**85**:732–752.

13. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., et al. Human-level control through deep reinforcement learning. *Nature* 2015;**518**(7540):529–533.
14. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:150902971* 2015;.
15. Rijken, T.. *DeepLight: Deep reinforcement learning for signalised traffic control*. Ph.D. thesis; Masters Thesis. University College London; 2015.
16. van der Pol, E.. *Deep reinforcement learning for coordination in traffic light control*. Ph.D. thesis; Masters Thesis. University of Amsterdam; 2016.
17. Li, L., Lv, Y., Wang, F.Y.. Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica* 2016;**3**(3):247–254.
18. Genders, W., Razavi, S.. Using a deep reinforcement learning agent for traffic signal control. *arXiv preprint arXiv:161101142* 2016;.
19. Casas, N.. Deep deterministic policy gradient for urban traffic light control. *arXiv preprint arXiv:170309035* 2017;.
20. El-Tantawy, S., Abdulhai, B., Abdelgawad, H.. Design of reinforcement learning parameters for seamless application of adaptive traffic signal control. *Journal of Intelligent Transportation Systems* 2014;**18**(3):227–245.
21. Mannion, P., Duggan, J., Howley, E.. An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In: *Autonomic Road Transport Support Systems*. Springer; 2016, p. 47–66.
22. Yau, K.L.A., Qadir, J., Khoo, H.L., Ling, M.H., Komisarczuk, P.. A survey on reinforcement learning models and algorithms for traffic signal control. *ACM Computing Surveys (CSUR)* 2017;**50**(3):34.
23. Sutton, R.S., Barto, A.G.. *Reinforcement learning: An introduction*; vol. 1. MIT press Cambridge; 1998.
24. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:160304467* 2016;.
25. Jones, E., Oliphant, T., Peterson, P., et al. SciPy: Open source scientific tools for Python. 2017.
26. Kapturowski, S.. Tensorflow-rl. 2017. URL `https://github.com/steveKapturowski/tensorflow-rl`.