Journal of Traffic and Transportation Management
Volume 1, No. 1 (2019) pp. 19-26

Evaluating Reinforcement Learning State Representations for Adaptive Traffic Signal Control

Wade Genders[a]*, Saiedeh Razavi[ab]

[a]*Department of Civil Engineering, McMaster University, Hamilton, Canada, L8S 4L8*
[b]*McMaster Institute for Transportation and Logistics, Hamilton, Canada, L8S 4L8*

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 and distributional Q-learning algorithms 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) using fully connected neural networks, but deep neural networks with high-resolution state representation achieve the best performance. These results indicate implementing reinforcement learning traffic signal controllers in practice can be accomplished with a variety of sensors (e.g., loop detectors, cameras, radar).

Keywords: *adaptive traffic signal control, deep reinforcement learning, intelligent transportation systems, applied machine learning, transportation simulation, neural networks*

1. Introduction

Vehicle congestion is a major problem in cities across the world [1]. 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 as an intelligent transportation system with many research studies detailing 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 [2] and the asynchronous advantage actor-critic (A3C) [3] and Categorical 51 atom (C51) [4] distributional Q-learning [5] algorithms to train and evaluate multiple adaptive TSC with different resolution state representations.

2. Related Work

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 [6–9]. Later research applied reinforcement learning methods to more realistic and complex traffic models [10–15]. Developments in machine learning have yielded deep reinforcement learning techniques [3, 16, 17] which have subsequently been applied for TSC [18–24]. Considering the aforementioned research and the extensive reinforcement learning TSC reviews [25–27], we identify

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

DOI: 10.5383/JTTM.01.01.003

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. Results from this research 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 [28]. A reinforcement learning agent learns a policy $\pi(s) = a$, mapping from states s to actions a, to achieve a goal, quantified as a reward r, 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 Eq.1:

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

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 transitioning the environment to 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. Reinforcement learning utilizes various types of value functions to develop the optimal policy. Many algorithms have been proposed to efficiently develop value functions using sampling-based techniques. The A3C algorithm is used to develop parameterized θ policy $\pi(a|s;\theta)$ (Eq.2) and state-value $V^{\pi}(s;\theta)$ functions (Eq.3). The agent develops a state-value function (critic), which estimates the expected return from a given state s_t, which is used to improve the policy (actor). The A3C algorithm is an on-policy reinforcement learning algorithm, developing the optimal policy directly.

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

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

The C51 algorithm [4] is an extension of the deep Q-network (DQN) algorithm [16], which develops a parametrized (state) action-value function $Q^{\pi}(s,a;\theta)$ (Eq.4).

$$Q^{\pi}(s,a;\theta) = E[G_t|s_t = s, a_t = a;\theta] \quad (4)$$

An action-value function estimates the expected return of taking action a_t from state s_t. DQN estimates an action-value's mean; C51 extends beyond the mean by modeling the distribution of the return incorporating the return's variance, depicted in Fig. 1. Like DQN, C51 is an off-policy, value-based reinforcement learning method, different from the A3C algorithm, which does not directly develop the optimal policy. Instead, the C51 algorithm can be used to develop the optimal policy by acting greedily with respect to the current state's action-value (i.e., choosing the action with the high value given the current state). Compared to policy-based reinforcement learning algorithms like A3C, C51 is more sample-efficient due to its use of an experience replay [30]. However, policy-based algorithms like A3C often have stronger convergence properties due to their direct development of the policy.

Both algorithms make use parametric function approximation in the form of neural networks. The parameters define the weights between neurons in the network, estimated through sampling, gradient based techniques [31–33].

3.2. Environment

The environment used to train the reinforcement learning adaptive TSC is the traffic microsimulator SUMO [2]. 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 N (λ, λ/10), displayed in Fig. 2.

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 × 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 previous 10 s interval. Given m incoming lanes, the A3C-Loop 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 V_l/k_j and vehicle lane queue $V_{l,,q}/k_j$. Given m incoming lanes, we denote the A3C-Queue 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 A3C-DTSE state is $s_t \in \mathbb{R}^{((L/c)\ m+|P\ |+1)t}$, where we use a history of the t = 2 most recent states. This state definition was first proposed using SARSA reinforcement learning [6] and has since been utilized in deep reinforcement learning TSC [18, 19, 21].
The C51 variant utilizes the highest-resolution state space, DTSE, to create a C51-DTSE TSC. The only difference between the A3C-DTSE and C51-DTSE states is that C51 stacks the t = 4 most recent states into each observation, similar to DQN [16].

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 = {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.

3.5. Rewards

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 at was in s_t. Many rewards have been proposed for reinforcement learning TSC (e.g., functions of throughput, queue, delay). The A3C's reward is defined in Eq.5 as change in cumulative delay:

$$r_t = D_{a_t} - D_{a_{t+1}} \tag{5}$$

Where D_{a_t}, $D_{a_{t+1}}$ represent the cumulative delay at the intersection when action at and a_{t+1} are taken. Cumulative vehicle delay D at time t is defined in Eq.6:

$$D_t = \sum_{v \epsilon V_t} d_t^v \tag{6}$$

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.
Preliminary work found the C51-DTSE performed poorly with the aforementioned change in delay reward. Further investigation found negative cumulative delay a suitable reward, defined in Eq.7:

$$r_t = -D_t \tag{7}$$

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.
An artificial neural network is chosen for its flexible function approximation capabilities. The A3C agent's neural network architecture 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 (i.e., next traffic phase). The number of neurons in each hidden layer for each state representation is equal to the cardinality of the input state; A3C-Loop and A3C-Queue hidden layers have 42 neurons and A3C-DTSE 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 multistep returns 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 for updating the state-value (Eq.8) and policy functions (Eq.9). Each agent periodically copies the global parameters θ as their local parameters θ'. The rewards are standardized before computing the return (i.e., $r_t \leftarrow (r_t - \mu_r)/\sigma_r$) and the gradient of the policy entropy $\epsilon_t = \beta\ \nabla_{\theta'} H(\pi(s;\theta'))$ is added to the parameter update for improved learning.

$$d\theta \leftarrow d\theta + \frac{\partial(\mathrm{Adv})^2}{\partial\theta'} \tag{8}$$

$$d\theta \leftarrow d\theta + \nabla_{\theta'} \log_e \pi(a|s;\theta')\,(\mathrm{Adv}) + \epsilon_t \tag{9}$$

The C51-DTSE uses a deep convolutional neural network architecture. The first layer receives the DTSE as input, with the following sequence of hidden convolutional and fully connected layers: 16 filters of size 4x4, 16 filters of size 3x3, 32 filters of size 2x2, a fully connected layer of 256 neurons and $|A|$ = 4 separate output heads of 51 fully connected neurons, representing each of the 4 action's return distributions. All hidden layers use ReLu activation functions except the final output heads, which uses softmax activation functions. The C51 algorithm updates using a categorical cross-entropy loss where the targets are discrete distributional returns. Readers interested in C51 algorithm details should consult the algorithm designer's original publication [4]. A decentralized acting, centralized learning architecture is used to train the C51-DTSE. Parallel actors explore distinct environments and send experiences (i.e., tuples of (s_t, a_t, r_t, s_{t+1})) to a centralized learner experience replay [34], each actor implementing a different ε-greedy exploration policy. The learner uniformly samples experience batches from the replay and computes parameter updates using online and target parameter sets. One important difference between the A3C and C51 implementations in this research is that A3C uses 32-step returns while C51 uses only 1-step returns.

4. Experiments

We subject each different state representation reinforcement learning TSC to $N = 100$ 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 Fig. 2. Training is conducted using a consumer i7 CPU with 8 parallel, asynchronous threads. 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 (i.e., random seeds) to ensure parity during testing. Throughput, delay and queue statistics are collected during testing simulations. During testing simulations, all network parameters are frozen; the A3C follows its learned policy and the C51 acts with a fixed exploration rate of 0.05.

The traffic microsimulator SUMO [2] is used for all simulations. Tensorflow [35], Keras [36], SciPy [37] and additional Python libraries [38, 39] 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.

All TSC models use the Adam optimizer [40] for training with a learning rate of 7.5×10^{-4} and an epsilon of 1×10^{6}. A batch size of 16 is used and both networks update after 100 experiences (i.e., online parameters are copied to target parameters for C51 and gradients are applied to global parameters for A3C).

Hyperparameters unique to C51 used in this research include distributional return maximum $V_{max} = 0.0$ and minimum $V_{min} = -10.0$ values. The experience replay has a maximum capacity of 1×10^{4} and is filled to capacity before updates begin. A visual representation of C51-DTSE TSC's distributional returns after training can be seen in Fig. 1.

5. Analysis & Discussion

Testing results are displayed in Table 2 and visually in Fig. 3. 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 must implement phases in a cycle.

Observing the traffic metrics collected, there appears to be little to no difference between the different state representations when using the A3C algorithm, however, the C51-DTSE TSC achieves the best performance overall. There is no difference in throughput or queue between A3C variants, however, the A3C-Loop exhibits the highest delay compared to the A3C-Queue and A3C-DTSE. This result is surprising to the authors, specifically how little difference there is between the different A3C state representations considering the spectrum of data resolution modeled.

The fact that the C51-DTSE outperformed all A3C variants is likely due to a combination of factors; the distributional return model and the difference in neural network architecture (i.e., deep convolutional versus shallow fully-connected). Although in theory A3C is described as on-policy, updating using asynchronous policy gradients, in practice policy lag can occur, making it slightly off-policy, diminishing the quality of the gradient updates and reducing learning performance. The A3C's behaviour is contrasted with C51's off-policy experience replay buffer, which is more robust while learning because the gradient updates are computed at the centralized learner. While the C51 algorithm with high-resolution state data achieves the best performance, the performance disparity is not as significant as has been observed in other reinforcement learning tasks, such as Atari video games [4, 41]. Low-resolution state data, as can be generated from loop detectors, still achieves considerable performance gains compared to traditional, Actuated TSC. These findings suggest common traffic sensors such as loop detectors would be sufficient to provide data for any parties interested in implementing reinforcement learning TSC in practice. However, parties interested in achieving the maximum performance should consider deep neural networks with high-resolution state data and a distributional return model.

6. Conclusion

We modeled adaptive TSC using the A3C and C51 reinforcement learning algorithms 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 delays and queues by approximately $10 - 20\%$, with no difference in throughput compared to low-resolution state representations.

Considering the successful combination of experience replay and distributional returns, many future areas of research exists. Off-policy, policy-based reinforcement learning methods have been successfully developed which combine the data efficiency of experience replay with direct policy development [42–44]. Additionally, improved algorithms have been developed for modeling distributional returns, displaying improved performance beyond C51 [41]. Any of these techniques could be used to further improve reinforcement learning TSC and are worth future investigation.

Acknowledgements

Find some source code related to this research at https://github.com/docwza/deep-rl-tsc.

References

[1] Cookson, G. (2018). INRIX Global Traffic Scorecard. Technical report, INRIX

[2] Krajzewicz, D., Erdmann, J., Behrisch, M., and 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.

[3] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In International Conference on Machine Learning, pages 1928–1937.

[4] Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. arXiv preprint arXiv:1707.06887.

[5] Watkins, C. J. and Dayan, P. (1992). Q-learning. Machine learning, 8(3-4):279–292. https://doi.org/10.1023/A:1022676722315

[6] Thorpe, T. L. and Anderson, C. W. (1996). Traffic light control using sarsa with three state representations. Technical report, Citeseer.

[7] Wiering, M. et al. (2000). Multi-agent reinforcement learning for traffic light control. In ICML, pages 1151–1158.

[8] Brockfeld, E., Barlovic, R., Schadschneider, A., and Schreckenberg, M. (2001). Optimizing traffic lights in a cellular automaton model for city traffic. Physical Review E, 64(5):056132. https://doi.org/10.1103/PhysRevE.64.056132

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

[10] Wiering, M., Vreeken, J., Van Veenen, J., and Koopman, A. (2004). Simulation and optimization of traffic in a city. In Intelligent Vehicles Symposium, 2004 IEEE, pages 453–458. IEEE. https://doi.org/10.1109/IVS.2004.1336426

[11] El-Tantawy, S., Abdulhai, B., and 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. https://doi.org/10.1109/TITS.2013.2255286

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

[13] Zhu, F., Aziz, H. A., Qian, X., and Ukkusuri, S. V. (2015). A junction-tree based learning algorithm to optimize network wide traffic control: A coordinated multiagent framework. Transportation Research Part C: Emerging Technologies, 58:487–501. https://doi.org/10.1016/j.trc.2014.12.009

[14] Jin, J. and Ma, X. (2017). A group-based traffic signal control with adaptive learning ability. Engineering applications of artificial intelligence, 65:282–293. https://doi.org/10.1016/j.engappai.2017.07.022

[15] Aslani, M., Mesgari, M. S., and Wiering, M. (2017). 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, 85:732–752. https://doi.org/10.1016/j.trc.2017.09.020

[16] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Humanlevel control through deep reinforcement learning. Nature, 518(7540):529–533. https://doi.org/10.1038/nature14236

[17] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

[18] Rijken, T. (2015). DeepLight: Deep reinforcement learning for signalised traffic control. PhD thesis, Master's Thesis. University College London, London, United Kingdom.

[19] Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In AAAI, pages 2094–2100.

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

[21] Genders, W. and Razavi, S. (2016). Using a deep reinforcement learning agent for traffic signal control. arXiv preprint arXiv:1611.01142.

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

[23] Wan, C.-H. and Hwang, M.-C. (2018). Value-based deep reinforcement learning for adaptive isolated intersection signal control. IET Intelligent Transport Systems, 12(9):1005–1010. https://doi.org/10.1049/iet-its.2018.5170

[24] Nishi, T., Otaki, K., Hayakawa, K., and Yoshimura, T. (2018). Traffic signal control based on reinforcement learning with graph convolutional neural nets. In 21st International Conference on Intelligent Transportation Systems. IEEE. https://doi.org/10.1109/ITSC.2018.8569301

[25] Genders, W. and Razavi, S. (2018). Evaluating reinforcement learning state representations for adaptive traffic signal control. Procedia computer science, 130:26–33. https://doi.org/10.1016/j.procs.2018.04.008

[26] El-Tantawy, S., Abdulhai, B., and 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. https://doi.org/10.1080/15472450.2013.810991

[27] Mannion, P., Duggan, J., and Howley, E. (2016). An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In Autonomic Road Transport Support Systems, pages 47–66. Springer. https://doi.org/10.1007/978-3-319-25808-9_4

[28] Yau, K.-L. A., Qadir, J., Khoo, H. L., Ling, M. H., and Komisarczuk, P. (2017). A survey on reinforcement learning models and algorithms for traffic signal control. ACM Computing Surveys (CSUR), 50(3):34. https://doi.org/10.1145/3068287

[29] Sutton, R. S. and Barto, A. G. (1998). Reinforcement learning: An introduction, volume 1. MIT press Cambridge.

[30] Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine learning, 8(3-4):293–321. https://doi.org/10.1007/BF00992699

[31] Linnainmaa, S. (1970). The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors. Master's Thesis (in Finnish), Univ. Helsinki, pages 6–7.

[32] Werbos, P. J. (1982). Applications of advances in nonlinear sensitivity analysis. In System modeling and optimization, pages 762–770. Springer https://doi.org/10.1007/BFb0006203

[33] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science. https://doi.org/10.21236/ADA164453

[34] Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Van Hasselt, H., and Silver, D. (2018). Distributed prioritized experience replay. arXiv preprint arXiv:1803.00933.

[35] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467

[36] Chollet, F. (2015). Keras. http://keras.com

[37] Jones, E., Oliphant, T., Peterson, P., et al. (2017). SciPy: Open source scientific tools for Python.

[38] Kapturowski, S. (2017). Tensorflow-rl. https://github.com/steveKapturowski/tensorflow-rl

[39] Flyyufelix. (2017). C51-ddqn-keras. https://github.com/flyyufelix/C51-ddqn-keras

[40] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[41] Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. (2017). Distributional reinforcement learning with quantile regression. arXiv preprint arXiv:1710.10044.

[42] Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and de Freitas, N. (2016). Sample efficient actor-critic with experience replay. arXiv preprint arXiv:1611.01224.

[43] Gruslys, A., Azar, M. G., Bellemare, M. G., and Munos, R. (2017). The reactor: A sample-efficient actor-critic architecture. arXiv preprint arXiv:1704.04651.

[44] Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. (2018). Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. arXiv preprint

[45] Liu, Y., Liu, L., and Chen, W.-P. (2017). Intelligent traffic light control using distributed multi-agent q learning. arXiv preprint arXiv:1711.10941. https://doi.org/10.1109/ITSC.2017.8317730

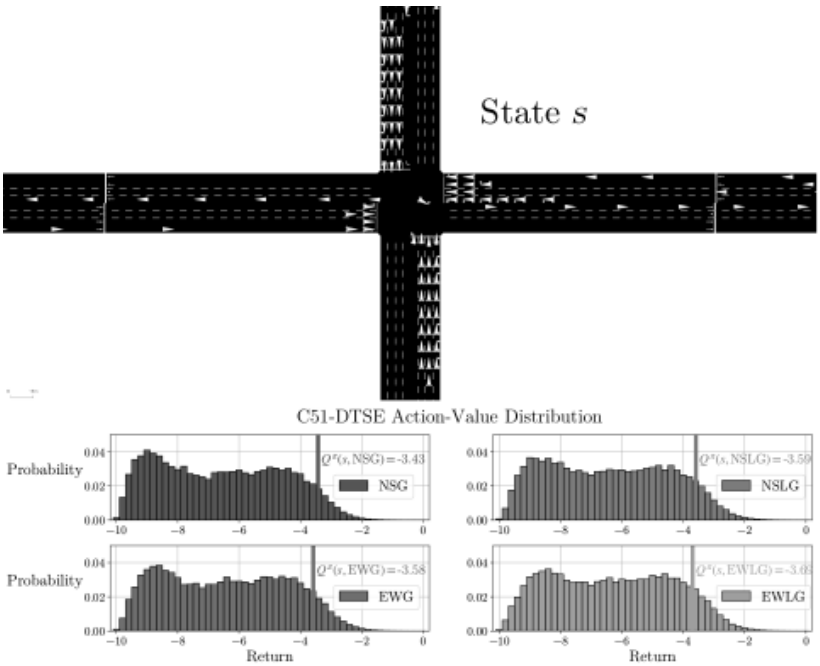

Fig. 1. C51-DTSE action-value distribution for the current state s in the SUMO traffic model. The solid vertical line in each subplot represents the expected value of the action-value's distribution. With reward defined as negative cumulative delay, the action value closest to 0.0 is optimal. Observing the simulator state s, the majority of vehicles are traversing North and South links. The action-value distributions reflect this fact, as the NSG action exhibits the highest action-value, indicating it is the optimal action.

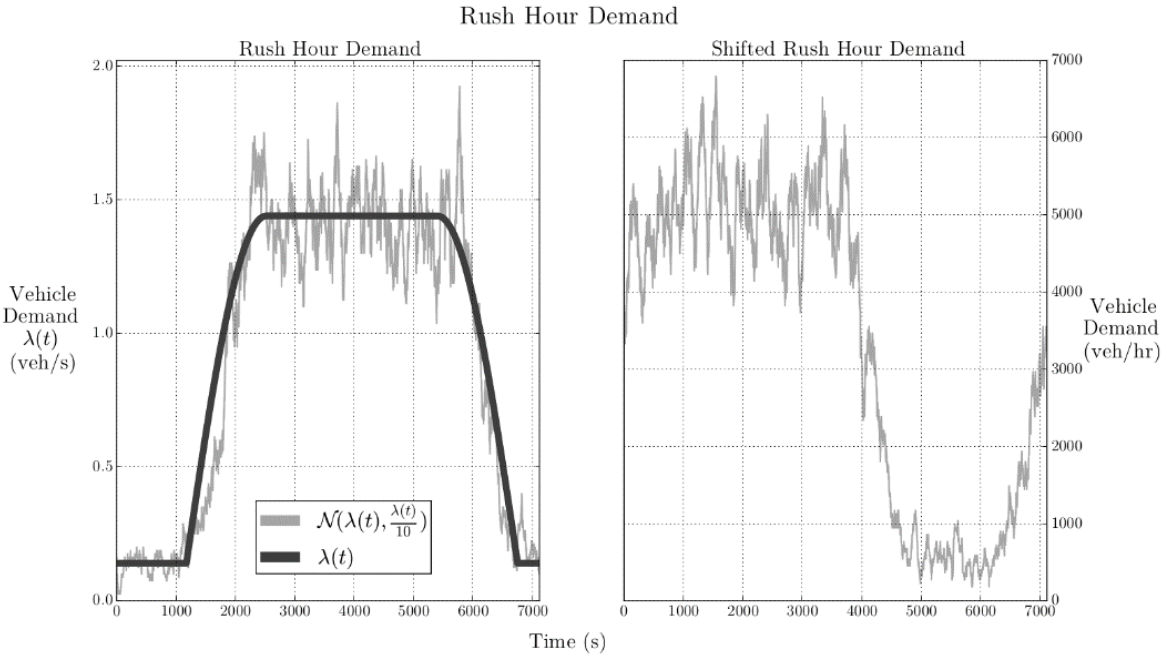

Fig. 2. Rush hour demand traffic scenario used for training (randomly shifted temporally) (right) and testing (fixed temporally) (left).

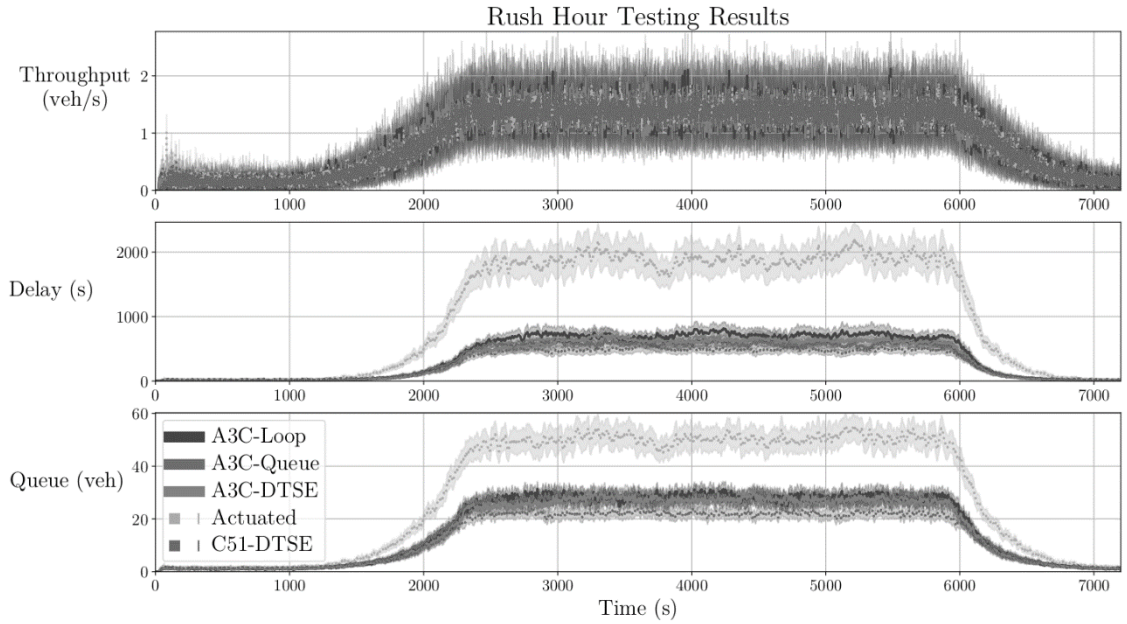

Fig. 3. Testing results comparing different agent and state definitions in a 2 hour simulation testing simulations. Each method's performance is estimated from 100 randomly generated (i.e., randomly seeded) testing simulations. Solid coloured lines represent mean values and shaded areas represent 95% confidence intervals.

Table 1. Traffic Signal Phase Information

			Turning Movements		
Action	NEMA Phases	Compass Directions	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

Table 2. Traffic Signal Controllers Estimated Performance

		(n=100, $\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}$)	
Traffic Signal Control Method	Total Throughput (veh/sim)	Total Delay (s/sim)	Total Queue(veh/sim)
A3C-Loop	(6 609, 86)	$(2.8 \times 10^6, 4.6 \times 10^5)$	$(1.2 \times 10^5, 0.8 \times 10^4)$
A3C-Queue	(6 612, 87)	$(2.3 \times 10^6, 2.5 \times 10^5)$	$(1.2 \times 10^5, 0.7 \times 10^4)$
A3C-DTSE	(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)	$(7.7 \times 10^6, 7.3 \times 10^5)$	$(2.2 \times 10^5, 1.2 \times 10^4)$
C51-DTSE	(6 610, 99)	$(2.0 \times 10^6, 2.8 \times 10^5)$	$(9.6 \times 10^4, 1.0 \times 10^4)$