Guidelines for Action Space Definition in Reinforcement Learning-Based Traffic Signal Control Systems

Maxime Treca,[1,2] **Julian Garbiso,**[1] **Dominique Barth**[2]
[1]Institut Vedecom, Versailles, France, [2]David Lab, UVSQ, Versailles, France
{maxime.treca, julian.garbiso}@vedecom.fr, dominique.barth@uvsq.fr

Abstract

Previous works in the field of reinforcement learning applied to traffic signal control (RL-TSC) have focused on optimizing state and reward definitions, leaving the impact of the agent's action space definition largely unexplored. In this paper, we compare different types of TSC controllers – phase-based and step-based – in a simulated network featuring different traffic demand patterns in order to provide guidelines for optimally defining RL-TSC actions. Our results show that an agent's performance and convergence speed both increase with its interaction frequency with the environment. However, certain methods with lower observation frequencies – that can be achieved with realistic sensing technologies – have reasonably similar performance compared to higher frequency ones in all scenarios, and even outperform them under specific traffic conditions.

Introduction

Traffic signal control (TSC) is a popular topic of study in the field of applicated reinforcement learning and real-time scheduling. On top of being an urban planning tool with important economic, social and environmental implications, TSC systems also provide a feedback loop decision framework that is very well suited to the study of reinforcement learning problems.

All TSC methods using reinforcement learning share a similar pattern (Mannion, Duggan, and Howley 2016): the agent first observes the current traffic and encodes it into a *state* using local information such as queue length, vehicle waiting time or traffic signal information. Given the current traffic state, the agent chooses an *action* that determines future traffic signals. Once the traffic signals are applied, the agent receives a *reward* denoting the quality of the previous action given the traffic state.

While most papers of the literature claim – rightfully so – that state and reward definitions are crucial in designing a reinforcement-learning-based controller for traffic signal control, only a few acknowledge the equally important role of action definitions, which are usually given as-is.

The aim of this paper is to study the impact of action definition in the performance of RL-TSC methods using Q-learning to route traffic. More particularly, the study focuses on the time interval between two successive decision points by comparing *step-based* actions – in which an agent can decide to switch or stay in the current traffic phase every $n \in \mathbb{N}$ seconds – and *phase-based* action in which an agent decides the entire phase duration before applying it. Simulation results show that step-based actions are more efficient than phase-based actions, but that this efficiency decreases as traffic heterogeneity increases within the network. These results also suggest that very small values of n do not yield better results than higher ones for step-based actions.

Related Works

Traffic signal control methods use traffic lights to ensure the safety and efficiency of a road network. These methods were traditionally divided in 3 generations (Gartner, Stamatiadis, and Tarnoff 1995), ranging from fixed methods dating from the 1950s to modern traffic flow prevision systems.

Reinforcement learning-based TSC methods (RL-TSC) – which are part of a newer, fourth generation of methods (El-Tantawy and Abdulhai 2012) – use machine learning algorithms to continuously learn from previous traffic signal decisions in an on-line fashion. Even though they have not yet been applied in real-world traffic control systems, learning-based methods are consistently outperforming fixed methods from previous generations in simulated environments (Yau et al. 2017).

Q-Learning (Watkins and Dayan 1992) is a on-line and off-policy reinforcement learning method that searches for an optimal agent policy by storing discounted state/action value estimates in a tabular fashion. Q-Learning, and RL-TSC methods more generally, have gained gradual popularity in the TSC literature since the early 2000s and the application of Q-Learning in a multi-agent setting (Wiering 2000). Most RL-TSC methods use Q-Learning as their learning algorithm, even though alternative methods such as Deep Q-Learning (Van der Pol and Oliehoek 2016) and actor-critic (Prashanth and Bhatnagar 2011) models have been investigated in the literature.

Apart from traditional issues arising from reinforcement learning problems (such as convergence, function approximation and the exploration/exploitation trade-off), the role of state (Genders and Razavi 2018) and reward (Touhbi et al. 2017) definitions has been studied in detail for RL-TSC methods. The role of action definitions has also been partly studied in MARLIN (El-Tantawy and Abdulhai 2012) which introduces a performance comparison for two different types of signal cycle definitions: fixed phasing scheme (FPS) – in which the controller decides whether to stay in or switch the current phase in a predefined cycle – and variable phasing scheme (VPS) in which the controller decides what phase to apply at the next decision point without any phase ordering constraints. Results showed that FPS yielded better performances under uniform traffic demands while VPS performed better under variable demands. Our paper tackles action definition in a more general sense as we do not consider whether phases are ordered or not within a signal cycle, but rather the time interval between two successive decision points. Indeed, current RL-TSC models either consider phase-based actions (Abdoos, Mozayani, and Bazzan 2011) or step-based actions (Prashanth and Bhatnagar 2011) without discussing the influence of these action intervals nor the respective strengths and weaknesses of both approaches.

Model

We consider a single intersection composed of 4 two-lanes approaches and of a traffic signal controller. Our model is in discrete time, where $t \in \mathbb{N}$ denotes the current system step.

Traffic Signals

The intersection features a traffic controller using a *signal cycle* composed of multiple *phases*, giving a right of way to specific approaches (i.e. roads) of the intersection.

Phases We consider a set of phases $\Psi = \{\psi_1, \psi_2, \psi_T\}$ that the controller can implement. The phase ψ_1 provides green lights to the northern and southern approaches of the intersection, allowing them to reach any other approach within the intersection. Similarly, the phase ψ_2 provides green lights to the eastern and western approaches of the intersection. Finally, a transition phase ψ_T is only composed of red lights and is used as a transition phases between green light phases ψ_1 and ψ_2.

Signal Cycle The controller routes traffic on the intersection using a signal cycle Φ, repeating phases in an endless pattern $\Phi = \psi_1, \psi_T, \psi_2, \psi_T, \psi_1, \dots$ in a fixed phasing scheme (FPS) fashion. Hence, all even-numbered phases of Φ are transition phases and green light phases ψ_1 and ψ_2 are alternating on all odd-numbered phases.

Phase Duration Each phase ϕ_i of Φ has an associated duration d_i in number of steps. This duration is bounded by minimum d_{min} and maximum values d_{max} ensuring safety and waiting requirements of the road network.

Controller Types

Given that the signal cycle uses a fixed phasing scheme, the controller can only influence traffic by repeatedly defining the lengths of green phases ψ_1 and ψ_2. There are two main ways phase durations can be set.

Phase-based controller A *phase-based* controller sets the entire phase duration at once. Upon giving a duration d_i to phase ϕ_i, the controller runs phase ϕ_i for d_i steps before automatically switching to phase ϕ_{i+1}.

Step-based controller A *step-based* (or n-step) controller chooses whether to extend the current phase by an additional n steps or whether to switch to the next phase of the signal cycle. The value of n is a fixed parameter depending on the controller. Hence, a n-step controller can extend a single phase multiple times as long as it satisfies the phase duration constraints stated above.

Learning Framework

Most reinforcement learning methods feature the same following learning pattern. An agent first observes a state s_t and then selects an action a_t depending on this state. After the action is applied, the system transitions to a new state $s_{t+\Delta}$. The agent then receives a reward r_t denoting the quality of its previous action selection. These three definitions – state, action and rewards – are fundamental in RL models.

Agent While most RL-TSC models often feature multiple agents learning concurrently (MARL), ours focuses on a single agent over a single intersection. This design decision ensures that our study of the effect of action definition on a RL-TSC controller learning and traffic routing abilities does not suffer from network topology or non-stationarity issues (Buşoniu, Babuška, and De Schutter 2010).

State The traffic state at step t is defined by a 4-tuple $s_t = \langle \phi_i, d_i, c_{1,t}, c_{2,t} \rangle$ composed of the active phase, ϕ_i, the duration d_i for which it has been active, and the congestion of the approaches associated with phases ψ_1 and ψ_2. For $n_{a,t}$ the number of vehicles on approach a at time t, the congestion value $c_{i,t}$ is given by

$$c_{i,t} = (\sum_{a \in \psi_i} n_{a,t}) \bmod 3$$

Action For a phase-based controller, the agent's action space is the $[0, d_{max} - d_{min}]$ interval, which represents the set of possible phase durations under time constraints. For a step-based controller, the agent's action space is $[0, 1]$, where 0 represents a phase extension of n steps and 1 represents a switch to the next phase of the signal cycle.

Reward The agent's reward at a given step t is equal to the difference between the cumulated waiting time of vehicles on its approaches during the last decision point and the cumulated waiting time at time t.

Learning Method Agents of our model use Q-Learning (Watkins and Dayan 1992) as a learning algorithm. The agent stores the quality of all the state/action couples it encounters in a Q-table. Given $\alpha \in [0, 1]$ and $\gamma \in [0, 1]$ which represent the learning rate and discount factor of the agent each couple of the Q-table is updated according to the following rule:

$$Q(s_t, a_t) \leftarrow (1-\alpha_t)Q(s_t, a_t) + \alpha_t(r_t + \gamma \max_a Q(s_{t+1}, a))$$

We use a learning rate is $\alpha = \max(1/v(s_t, a_t), 0.05)$ where $v(s_t, a_t)$ is the number of times the agent visited the state/action couple (s_t, a_t). On top of giving less weight to new observations of often visited couples, this value of α ensures that the convergence properties of Q-Learning are respected (Sutton and Barto 1998).

Agent Policy The agent follows a ε-greedy policy. During a decision point, the agent will choose the action associated with the highest Q-table value with probability $(1 - \varepsilon)$, or a random action with probability ε. For a simulation iteration i, the parameter is set to $\varepsilon = \max(e^{-0.05i}, 0.05)$, which ensures high exploration during the first simulation iterations and high exploitation of good actions later on.

Results

Our aim is to compare different action definitions in order to provide guidelines for RL-TSC models. In order to do so, we compare different controllers – a phase-based agent, multiple n-step agents as well as a fixed controller – in a simulated traffic environment featuring different traffic situations.

Experimental Setup

The TSC methods and learning algorithms are implemented in Python, and communicate with the SUMO simulator (Krajzewicz et al. 2012) through the TRACI API. The results presented in this paper are averaged over 5 complete repetitions using different random seeds.

Simulation Runs A simulation scenario lasts for 10,000 steps, representing approximately three hours of real-time traffic. One step in SUMO is equivalent to one second of real simulated time. Each scenario goes through a hundred iterations, with the agent's Q-tables transferring between each iteration. Minimal and maximal phase durations, d_{min} and d_{max} are set to 5 and 45 steps respectively. All transition phases ψ_T have a duration of 5 steps.

Traffic generation Traffic demand is generated using two parallel Poisson processes. One Poisson process of parameter λ is uniformly distributed, meaning that each vehicle arrival is equally likely to happen on every approach of the intersection. The second Poisson process, using parameter τ, only assigns arrivals to the northern and southern approaches of the intersection. By supposing that the overall vehicle arrival rate is constant (i.e. $\lambda + \tau = 0.5$), we can hence define multiple different traffic patterns, ranging from uniform situations (low values of τ) to arterial/side street patterns (high values of τ).

Performance Evaluation

We consider that the learning method has converged when the difference between the moving average of last five performance measures is lower than a threshold of 10%. Once convergence, occurs, agent performance is measured as the average of total waiting time of vehicles over all subsequent iterations. An overview of simulation results per agent and traffic flow types can be seen on Table 1.

τ	Fixed	Phase	Step (Best)	Step (Worst)
0.0	3.617	2.672	2.053	2.473
0.1	4.070	2.746	1.956	2.595
0.2	4.603	3.070	1.977	2.570
0.3	7.773	4.582	2.032	2.531
0.4	6.807	5.773	2.088	2.216
0.5	18.329	3.240	1.994	2.473

Table 1: Average vehicle waiting time after convergence per agent type and traffic parameter τ (in 10^3 seconds).

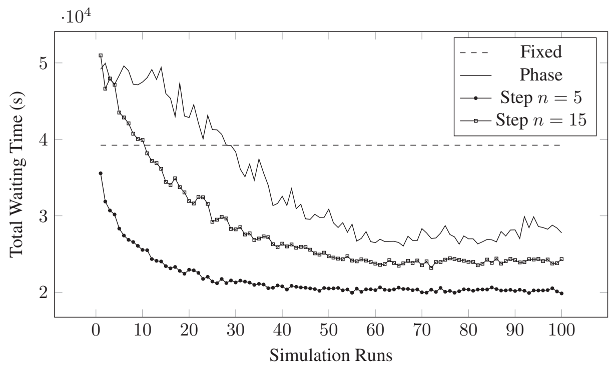

Figure 1: Vehicle waiting time per agent type, $\tau = 0.10$.

Fixed Method Unsurprisingly, the performances of the fixed method decrease as traffic demand becomes increasingly unidirectional through parameter τ. Both learning controllers beat the fixed method by a significant margin in every tested traffic scenario.

Phase-based Method Table 1 shows us that the phase-based method is less efficient than any n-step method for any value of τ. However, the performance improvements on Figure 1 throughout simulation iterations imply that the lack of performance of the phase-based methods is not due to its inability to learn from its environment, but is rather due to the fact that they are by nature less flexible than step-based ones. Indeed, the single-decision-point nature of phase-based methods has negative consequences on their performances: they cannot adjust to changing traffic situations within a single phase, as opposed to step-based methods that can re-assess the necessity to switch phases given the current traffic conditions multiple times within the same phase. The lesser amount of interactions with the environment could reduce convergence speed for the phase-based agent, which in turn would lower its performance. This hypothesis raises an important question, which is whether an increase in the frequency of an agent's interactions with its environment necessarily results in an improvement of its performance.

Step-based Methods On top of performing better than other methods for all tested traffic scenarios, results of the n-step methods listed in Table 1 show that these methods are

τ	$n=1$	$n=5$	$n=10$	$n=15$	$n=20$
0.0	0	4.86	10.29	22.37	27.81
0.1	0	4.17	7.20	23.96	30.15
0.2	0	0.45	5.49	22.68	31.03
0.3	3.04	0	4.38	11.02	26.39
0.4	9.53	0	2.74	11.09	7.84
0.5	22.12	0.56	0	13.60	3.33

Table 2: Percentage difference with respect to the optimum average vehicle waiting time (marked as 0) for step-based methods by action interval value n and traffic scenario τ

extremely robust as they reach similar performance levels for all values of τ where other methods perform worse when demand uniformity decreases. These observations clearly show that step-based RL-TSC methods perform better than their phase-based counterparts.

Guideline 1 Step-based methods should be chosen for RL-TSC methods under any traffic situation.

Step Action Interval

We now investigate the influence of parameter n on n-step agent performances in order to establish the optimal value of n and whether this optimality changes according to different traffic scenarios.

Optimal Action Interval Value The differences in performance between multiple n-step methods listed on Table 2 allow us to make two important observations. First, the optimal value of the action interval parameter n changes depending on the traffic demand, which implies that higher interaction frequency of the agent with its environment does not necessarily results in better performances. Second, there seems to be a clear relationship between the optimal value of n and the value of τ, since n-step methods using higher values of n perform gradually better when unidirectional traffic increases.

Guideline 2 Very short intervals between decision points are preferable for intersections with a uniform traffic demand while slightly longer intervals are preferable for intersections with skewed demand.

Convergence Figure 2 shows us that if all n-step methods eventually learn to assign longer green-light durations in highly skewed traffic situations, methods with lower values of n have a somewhat delayed and weaker response. Preliminary results lead us to believe that these convergence issues are at least partly due to the ε-greedy policy of the agent. Indeed, every time an n-step agent has to pick an action, it can randomly choose to switch the current phase with probability $\varepsilon/2$. Since ε has a very high value in the first simulation iterations and since agents with small action intervals are, by design, choosing actions very frequently, they are very unlikely to apply high phase duration values at first. This would explain why the agent $n=1$ on Figure 2 only starts converging on later iterations. These results lead us to recommend using values of n between $n=5$ and $n=10$, as

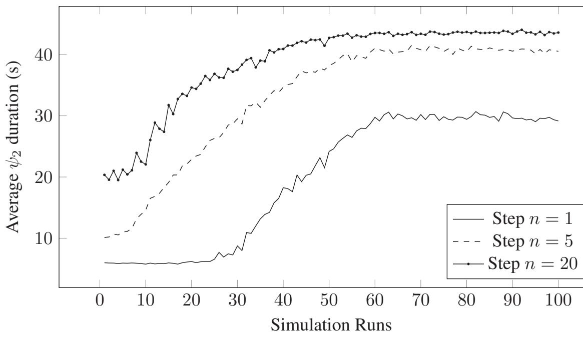

Figure 2: Evolution of phase ψ_2 average duration per simulation iteration, $\tau=0.50$.

they are either optimal or close to optimal in all traffic scenarios (see Table 2) and do not suffer from the convergence issues discussed before.

Feasibility Another added benefit of using longer action intervals for n-step agents is that they are better suited for using existing economically and technically competitive traffic sensing technologies. If measurement intervals of one second or less had been deemed optimal, it would have forced a trade-off between technical and economical feasibility and system performance, which is fortunately shown not to be the case.

Guideline 3 Defining longer intervals between successive decision points (from 5 to 10 seconds) yields satisfactory to optimal results for step-based agents. These values are also better suited for using existing economically and technically competitive traffic sensing technologies.

NEMA-type Signal Cycles We finally test our previous guidelines on a NEMA-style signal cycle composed of four green phases $\Psi = \{\psi_{1,A}, \psi_{1,B}, \psi_{2,A}, \psi_{2,B}\}$ where straight/right-turn phases (A) and left turn phases (B) are represented separatly for N/S (1) and E/W (2) approaches. Using this extended set of phases in a fixed phasing scheme globally yields results similar to the original case.

Conclusion

We investigated different RL-TSC action definitions in order to provide guidelines to design optimal TSC controllers. Simulation results lead us to recommend using step-based controllers instead of phase-based ones for traffic signal control. Moreover, we have shown that the optimal value of the action step parameter n of a step-based agent is dependent on the shape of the traffic demand on the intersection in which they are implemented. Finally, our simulations suggest that using action intervals between 5 and 10 seconds guarantee acceptable performances for any traffic situation and have the advantage of being better suited for using existing economically and technically competitive traffic sensing technologies. Future works include exploring the trade-off between shorter action intervals and convergence efficiency as well as an extension of our model in a multi-agent setting.

References

Abdoos, M.; Mozayani, N.; and Bazzan, A. L. 2011. Traffic light control in non-stationary environments based on multi agent q-learning. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, 1580–1585. IEEE.

Buşoniu, L.; Babuška, R.; and De Schutter, B. 2010. Multi-agent reinforcement learning: An overview. In *Innovations in multi-agent systems and applications-1*. Springer. 183–221.

El-Tantawy, S., and Abdulhai, B. 2012. Multi-agent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc). In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, 319–326. IEEE.

Gartner, N. H.; Stamatiadis, C.; and Tarnoff, P. J. 1995. Development of advanced traffic signal control strategies for intelligent transportation systems: Multilevel design. *Transportation Research Record* (1494).

Genders, W., and Razavi, S. 2018. Evaluating reinforcement learning state representations for adaptive traffic signal control. *Procedia computer science* 130:26–33.

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.

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*. Springer. 47–66.

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

Sutton, R. S., and Barto, A. G. 1998. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Touhbi, S.; Babram, M. A.; Nguyen-Huu, T.; Marilleau, N.; Hbid, M. L.; Cambier, C.; and Stinckwich, S. 2017. Adaptive traffic signal control: Exploring reward definition for reinforcement learning. *Procedia Computer Science* 109:513–520.

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

Watkins, C. J., and Dayan, P. 1992. Q-learning. *Machine learning* 8(3-4):279–292.

Wiering, M. 2000. Multi-agent reinforcement learning for traffic light control. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, 1151–1158.

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.