Parallel Reinforcement Learning with State Action Space Partitioning

Patrick Mannion P.MANNION3@NUIGALWAY.IE
Jim Duggan JIM.DUGGAN@NUIGALWAY.IE
Enda Howley ENDA.HOWLEY@NUIGALWAY.IE
Discipline of Information Technology, National University of Ireland Galway

Editors: Alessandro Lazaric, Mohammad Ghavamzadeh and Rémi Munos

Abstract

Parallel Reinforcement Learning (PRL) is an emerging paradigm within Reinforcement Learning (RL) literature, where multiple agents share their experiences while learning in parallel on separate instances of a problem. Here we propose a novel variant of PRL with State Action Space Partitioning (SASP). PRL agents are each assigned to a specific region of the state action space of a problem, with the goal of increasing exploration and improving learning speed. We evaluate our proposed approach on a realistic traffic signal control problem, and prove experimentally that it offers significant performance improvements over a PRL algorithm without SASP.

Keywords: Reinforcement Learning, Parallel Learning, Multi Agent Systems, Intelligent Transportation Systems, Adaptive Traffic Signal Control, Smart Cities

1. Introduction

In Parallel Reinforcement Learning (PRL), multiple agents share their experiences while learning in parallel on separate instances of a problem. Here we propose a novel variant of PRL with State Action Space Partitioning (SASP). We divide the state action space into multiple partitions, and PRL agents are assigned to explore each specific region with the goal of increasing exploration and improving learning speed. Generally, new PRL approaches are evaluated on traditional abstract problem domains (e.g. gridworld); however, here we evaluate our proposed algorithm using a realistic Traffic Signal Control (TSC) problem. In Reinforcement Learning for Traffic Signal Control (RL-TSC), an agent learns to control the signal switching sequence at a junction, with the goal of minimising delays.

Our contributions are as follows: 1) a novel method of Parallel Reinforcement Learning with State Action Space Partitioning is developed; 2) we evaluate the proposed method experimentally on a complex and realistic TSC problem; 3) we prove that using SASP can significantly increase learning speed and exploration compared to a PRL approach without SASP; 4) we discuss specific implementation requirements and future work.

The remainder of this paper is structured as follows: the second section discusses related research, while the third section describes our Parallel Reinforcement Learning architecture. The following two sections present the design of our experimental set up along with our experimental results. In the final section, we conclude with a discussion of our findings and our plans to extend this work further in the future.

2. Related Research

2.1. Reinforcement Learning

In Reinforcement Learning, an agent learns how to behave by a process of continuous interaction with its environment, generally without any prior knowledge of the problem domain. The agent receives a reward signal r based on the outcomes of previously selected actions, and by comparing stored estimates of r for individual state action pairs (Q values) the agent can decide which action is most appropriate to select when in a particular state.

Markov Decision Processes (MDPs) have become the de facto standard when formalising problems involving learning sequential decision making (Wiering and van Otterlo, 2012), and RL problems are generally modelled as an MDP. An MDP is defined by a tuple $< S, A, T, R >$, where R is the reward function, S is the set of all possible system states, A is the set of actions available to the agent, and T is the transition probability function. Selecting action $a \in A$ when in state s $\in$ S will cause the environment to transition into a new state $s' \in S$ with probability $T(s, a, s') \in$ (0,1), and give a reward $r = R(s, a, s')$.

RL algorithms are divided into two main categories: they are either model-based (e.g. Dyna, Prioritised Sweeping), or model-free (e.g. Q-Learning, SARSA). In the case of model-based approaches, agents attempt to learn the transition function T, and then T can be used when making action selections. This may be problematic, considering that T may be difficult to learn in complex problem domains. Model-free approaches instead rely on sampling the underlying MDP directly in order to gain knowledge about the unknown model. Thus, exploration is necessary for a model-free learner to gain the required knowledge about its environment, and the exploration vs exploitation dilemma discussed above must be balanced appropriately. The ϵ-greedy action selection strategy is an approach commonly used to obtain the required balance, where the highest valued action for a particular state is chosen with probability $1 - \epsilon$, or a random action is chosen with probability ϵ.

Q-Learning and SARSA are two of the most commonly used model-free RL algorithms. Q-Learning is an off-policy, model-free learning algorithm that has been proven to converge to the optimum action-values with probability 1 so long as all actions are repeatedly sampled in all states and the action-values are represented discretely (Watkins and Dayan, 1992). In Q-Learning, Q values are updated according to the equation below:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha(r_t + \gamma Q_{\max_a}(s_{t+1}, a) - Q_t(s_t, a_t)) \tag{1}$$

Here the learning rate $\alpha \in [0, 1]$ determines the degree to which Q values are updated at each time step t. The discount factor $\gamma \in [0, 1]$ controls how the agent regards future rewards. A low value of γ results in an agent which is myopic, while higher values of γ make the agent more forward looking.

2.2. Parallel Reinforcement Learning

Kretchmar (2002) describes a Parallel Reinforcement Learning (PRL) algorithm, which allows multiple agents to learn in parallel on the same task while sharing experience. A multi-armed bandit problem was used to test the proposed approach with varying numbers of parallel learning agents. The experimental results clearly show that the PRL approach

outperforms a single learner, while also demonstrating a reduction in the time taken to converge to an optimal policy with each additional parallel learner added.

Kushida et al. (2006) present a comparative study of three different parallel RL implementations, two of which were based on Q-Learning, while the third was based on fuzzy Q-Learning. The algorithms were tested on a gridworld domain, and the experimental results showed a clear performance benefit due to multiple agents learning in parallel.

Grounds and Kudenko (2008) developed a Parallel Reinforcement Learning approach which is capable of solving single agent learning problems in parallel on a cluster. This algorithm is based on SARSA, and learners share value function estimates which are represented by linear function approximators. By making use of the Message Passing Interface, agents asynchronously transmit messages containing their learned weight values over the cluster network. The experimental results demonstrated that the use of PRL techniques can reduce the time taken to compute good policies in single agent learning tasks.

A parallel framework for Bayesian Reinforcement Learning was developed by Barrett et al. (2014). In this model-based RL approach, the authors apply their algorithm to learn state transition probabilities for a given problem domain without any prior knowledge. The PRL approach was evaluated on two different experimental scenarios: a cloud resource allocation problem based on real world user demand data, and a stochastic gridworld domain. The proposed approach significantly improved convergence times in both test scenarios, using Kullback-Liebler Divergence as a metric. The benefits of PRL were also observed to scale well when additional PRL agents were added.

Mannion et al. (2015c) proposed Parallel Learning using Heterogeneous Agents. In this architecture, multiple PRL agents learn approximate knowledge about the problem domain using restricted action sets during a pre-training period, before transferring their combined knowledge to a superior agent which then learns using the full action set. The approach was evaluated using several gridworld domains, and was found to offer a statistically significant reduction in the number of steps taken to goal compared to a standard Q-Learner.

3. Parallel Reinforcement Learning with State Action Space Partitioning

There are two types of agents in our PRL implementation: Master, and Slave, both of which are based on Q-Learning. A single Master agent may be used to solve a problem on its own, or with the aid of one or more Slave agents. The Master and Slave agents learn simultaneously on different instances of the same problem (or similar, relevant problems). The Slave agents impart their experience to the Master Agent by means of a shared experience pool, in the form of a global Q matrix, as shown in Fig. 1. The Master and Slave agents each have their own instance of the problem environment to learn on, and at each timestep the knowledge learned by all agents is synchronised in the global Q matrix, in accordance with Equation 1. The Master agent may then draw on this experience to aid its action selection in the next timestep. Experimental results are measured from the instance of the problem environment owned by the Master agent in all of our tests. Each Slave agent selects actions using the ϵ-greedy strategy, where a random action is chosen with probability ϵ, or the action with the best expected reward is chosen with the remaining probability $1 - \epsilon$. The value of ϵ is set to 0.05 for all Slave agents in these experiments. This value of ϵ promotes exploitation of the knowledge the agent has gained, while still allowing for

sufficient exploration. The Master agent selects actions based on a pure greedy policy i.e. it always chooses the action with the maximum Q-value for a given state. This allows the quality of the policy learned by the PRL algorithm to be measured more accurately during testing, as the effect of random action selections is eliminated.

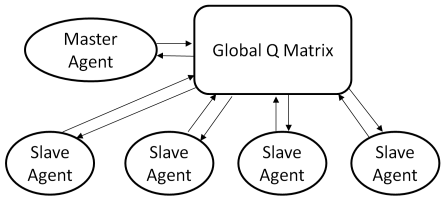


Figure 1: Agents share their experience via a global Q matrix

Here we extend our previous work on PRL for Traffic Signal Control (see Mannion et al. (2015b)) by introducing State Action Space Partitioning. Rather than using homogeneous Slave agents which learn on the entire state action space of the problem, here we partition the state action space of the problem domain among the Slave agents, with each learning about a specific subset of the entire problem domain. The goal of this approach is to provide more focused exploration and increase learning speed, while also reducing duplication of learning among Slave agents by ensuring that specific agents are responsible for sampling each subset of the state action space.

3.1. Agent Design

Here we use the same state, action and reward definitions for a traffic signal control agent as used by El-Tantawy et al. (2013), so our PRL method can be considered to be an extension of this approach. For a complete discussion of the applications of RL to TSC problems, we refer the interested reader to a review paper by Mannion et al. (2015a).

The environmental state is defined as a vector of dimension $2 + P$, where P is the number of phases at the junction, shown formally in Equation 2 below. The first two components in the state definition are the index of the current phase (P_c) and the elapsed time in the current phase (PTE), while the remaining P components represent the queue lengths (QL_i) for each phase at the junction.

$$s = \left[P_c, \quad PTE, \quad QL_0, \quad ..., \quad QL_n \right] \tag{2}$$

The maximum number of queueing vehicles considered is limited to 20, and the maximum phase elapsed time considered is limited to 30 seconds. Imposing these limits reduces the number of possible environmental states considered by an agent. A vehicle is considered to be queueing if its approach speed to the junction is less than 10 km/hr. Limiting the number of queueing vehicles about which an agent knows to 20 adds further realism to our experiments, as in practice it would be prohibitively complex and expensive to detect queueing vehicles along the entire length of the approach lane.

At each time step t, the actions available to the agents are: to keep the currently displayed green and red signals, or to set a green light for a different phase. To eliminate unreasonably low durations from consideration, phases are subject to a minimum length of 5 seconds. Agents are free to extend the current phase or switch to the next phase as they see fit, and there is no fixed cycle length. If an agent decides to switch phases, an amber signal is displayed for 3 seconds, followed by an all red period of 2 seconds, followed by a

green signal to the next phase. This adds greater realism as it accounts for lost time due to phase switching, along with reducing the chances of vehicle collisions occurring.

The reward function used by all agents is shown in Equation 3 below. When an agent selects an action a in a given state s and transitions to a resultant state s', the reward received is defined as the difference between the current and previous cumulative waiting times (CWT) of vehicles queueing at the junction. Therefore actions that decrease the cumulative waiting time receive a positive reward, while actions that increase the cumulative waiting time incur a negative reward (or penalty).

$$R(s, a, s') = CWT_s - CWT_{s'} \tag{3}$$

We test two different PRL algorithms; one with homogeneous Slave agents (Mannion et al., 2015b), and the version using State Action Space Partitioning (PRL-SASP) which we have proposed here. In the version with SASP, the state action space is partitioned equally among the Slave agents by restricting the maximum and minimum phase lengths that an agent is allowed to use, and the number of partitions depends on the number of Slave agents when using SASP. Traffic Signal Control is a non-deterministic problem domain; i.e. the next environmental state s' cannot be predicted using just the previous state s and selected action a. Therefore, it is not possible to directly assign specific ranges of states to each Slave agent. Partitioning based on states is more straightforward in simple problem domains, e.g. in a deterministic gridworld the total area could be divided into subregions for each Slave agent to explore. By partitioning the TSC problem based on actions (i.e. limiting a Slave agent's action selections based on phase length) it is possible for individual Slave agents to gain experience of different regions of the state space. We aim to prove that SASP can both improve learning speed and learn more about the problem by taking a more focused approach to exploration.

4. Experimental Design

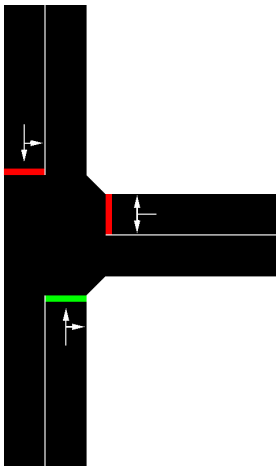


Figure 2: T junction used for testing

We have based our experimental setup around the microscopic traffic simulation package SUMO (Simulation of Urban MObility) (Krajzewicz et al., 2012), and agent logic is defined in our external framework, which is implemented in Java. The simulation timestep length is set to 1 second for all experiments. Each RL agent is associated with a specific simulation instance, and can only interact with and measure parameters in that instance. We use the TraaS library [1] to make simulation parameters available to the agents, and also to send signal switching instructions from the agents back to SUMO.

1. TraaS: TraCI as a Service, http://traas.sourceforge.net/cms/

All agents begin each experiment with their Q values for each state action pair initialised to zero. The values used for the learning rate α and the discount factor γ are 0.08 and 0.8 respectively. All learning agents in our experiments use these values of α and γ.

Our PRL approach is evaluated experimentally using an isolated T junction, with three intersecting two-way streets (shown in Fig. 4). The traffic demand D (measured in vehicles per hour) arriving at the junction is generated using a step function. This demand step function is comprised of a base flow b, episode length e, step demand increase h, and a step interval i. Thus the demand at any time t into a particular episode can be calculated according to Equation 4 below:

$$D(t) = \begin{cases} b + \dfrac{h \times t}{i} & t < \dfrac{e}{2} \\ b + h \times \left(\dfrac{e}{2 \times i} - 1 - \dfrac{t - 0.5 \times e}{i} \right) & \text{otherwise} \end{cases} \tag{4}$$

The increase in demand due to this function is computed at intervals equal to i. This time-varying traffic demand presents a more challenging flow pattern for the agents to control, compared to a constant hourly demand definition. The step function aims to emulate a peak in demand levels similar to a morning or evening rush hour. Agents are trained on this junction for 50 two hour training episodes, and the same demand step function is repeated during each episode. There are 6 possible routes through the junction, and three phases: North, East and South. The base flow is set to 1000 veh/hr, rising by 100 veh/hr every 15 minutes to a peak of 1300 veh/hr, before returning to 1000 veh/hr. Flows entering the junction are split in the following ratios: North 20%, East 30%, South 50%. The demand variation during a training episode is shown graphically in Fig. 3(a).

We first test a single RL agent as a benchmark. This single agent approach is similar to agents used by other researchers in RL-TSC, see e.g. El-Tantawy et al. (2013); Mannion et al. (2015d). The standard PRL algorithm is tested using 2, 3 and 4 agents, while our proposed PRL-SASP algorithm is tested using 3 and 4 agents.

5. Experimental Results and Discussion

We evaluated our approach by measuring the following parameters for each experiment: Average Waiting Times (AWT), Average Queue Lengths (AQL), and Average Number of States Visited (ANSV). We measured the values for the first two parameters from the SUMO instance associated with the Master agent using a customised data collection framework, while Number of States Visited was measured directly from the global Q matrix. The values reported for the first two parameters are the averages for each episode, while for ANSV the actual value at the end of each episode is reported. Experimental results are plotted in Figs. 3(b) to 3(d) and are summarised in Table 1. Percentages calculated in Table 1 refer to the improvement in performance versus a single agent approach. The results reported for each experiment are the average of 10 statistical runs.

In general, the plots of AWT (Fig. 3(b)), AQL (Fig. 3(c)) and ANSV (Fig. 3(d)) show improvements in learning speed when additional PRL agents are added for both standard PRL and PRL-SASP. When using 3 agents in parallel, there is practically no difference in the learning speed of standard PRL and PRL-SASP. However, when 4 agents are used in

parallel the state action space partitions are smaller, and in this case we see a noticeable difference between the learning speeds of standard PRL and PRL-SASP, especially in the plots of AWT and ANSV.

PRL-SASP with 4 agents has learned the best final policy at the end of the 50 episode training time, with a 37% reduction in AWT compared to a single agent. By contrast, standard PRL with 4 agents has only managed a 30% AWT improvement over a single agent in the same training time. A considerable improvement in ANSV is also evident in this case; the standard PRL approach has increased ANSV by 98%, whereas PRL-SASP has increased ANSV by 120% over the same training duration. Here there is an obvious benefit to using PRL-SASP, as many more of the possible states in this problem domain have been explored. These improvements in AWT and ANSV were found to be statistically significant when tested using two-tailed t-tests with $\alpha = 0.05$.

When using 3 agents, standard PRL and PRL-SASP have practically identical performance. This shows that PRL-SASP needs to be used with an adequate number of partitions to see the benefit of the partitioning effect. Using 3 partitions (a total of 4 agents) in this problem domain was found to offer good performance. Experimentation would be required to select a suitable number of partitions for other problem domains. Here we partitioned the state action space equally among the agents; however, more important or promising parts of the state action space could potentially be put into smaller partitions to ensure quick exploration of these areas. These types of measures are a way of using the system designer's knowledge of the problem domain in order to improve agent performance.

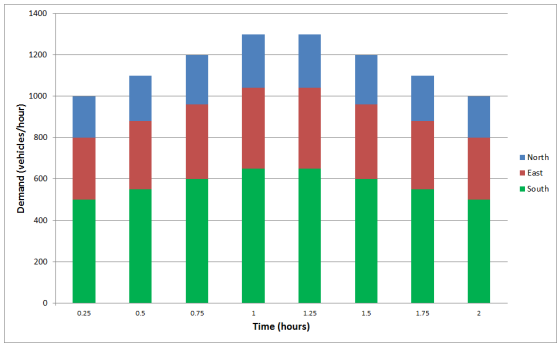
Table 1: Summary of Experimental Results (Average over Final 10 Episodes)

	Waiting Time			Queue Length			Number of States Visited		
	μ	% Reduction	σ	μ	% Reduction	σ	μ	% Increase	σ
1 agent	76.01	-	1.65	11.78	-	0.34	2340.72	-	124.3
2 agents PRL	57.74	24.03	2.01	9.02	23.44	0.24	3938.78	68.27	137.34
3 agents PRL	52.93	30.36	1.49	8.57	27.21	0.23	4360.53	86.29	159.65
3 agents SASP	53.29	29.90	1.60	8.67	26.42	0.25	4397.36	87.86	103.68
4 agents PRL	52.55	30.86	0.61	8.59	27.08	0.13	4630.03	97.80	97.11
4 agents SASP	47.52	37.47	1.11	8.30	29.52	0.19	5149.48	119.99	153.09

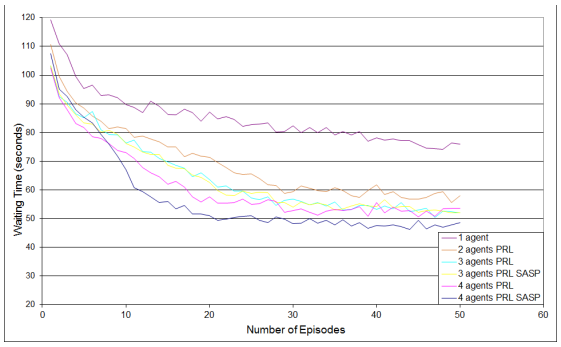
6. Conclusions and Future Work

In this paper, we have presented a novel variant of Parallel Reinforcement Learning, where the state action space is partitioned using domain knowledge. We have evaluated our approach experimentally using a realistic Traffic Signal Control problem, and proven that PRL-SASP can increase exploration and learning speed when compared to a standard PRL implementation. PRL-SASP using 4 agents was also found to learn a significantly better policy in the given training time, with a statistically significant reduction in average waiting times compared to PRL without SASP.

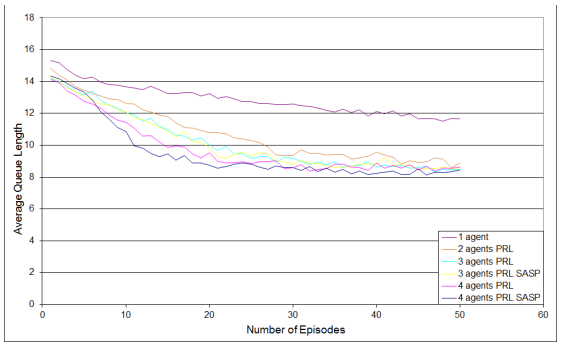
This paper has shown that PRL-SASP is a viable technique to use when learning in complex problem domains, although domain knowledge and experimentation is required in order to select the best partitioning strategy and optimum number of PRL agents. Particularly



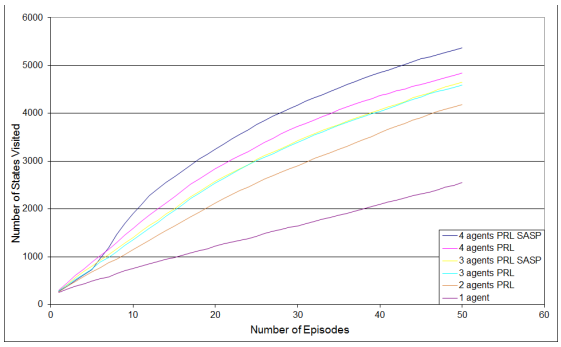
(*a*) Demand over two hour episode



(*b*) Waiting Times



(*c*) Queue Lengths



(*d*) Number of States Visited

Figure 3: Experimental Plots

noteworthy is its increased exploration when compared to a standard PRL implementation, considering that both approaches have almost identical computational complexity.

In the future, we plan to evaluate this technique further in alternative problem domains, including cloud resource allocation and smart grids. Each new application domain will present challenges in terms of determining a good partitioning strategy. Currently partitioning strategies must be designed manually using human knowledge of the problem domain, however the possibility exists to explore automated partitioning strategies in future work. We also plan to test PRL-SASP in a more complex Multi Agent variant of the domain used in this paper, with multiple signalised junctions based on real world traffic networks and demand data.

Acknowledgments

The primary author acknowledges the financial support provided to him by the Irish Research Council, through the Government of Ireland Postgraduate Scholarship Scheme.

References

Enda Barrett, Jim Duggan, and Enda Howley. A parallel framework for bayesian reinforcement learning. *Connection Science*, 26(1):7–23, January 2014. ISSN 0954-0091.

Samah El-Tantawy, Baher Abdulhai, and Hossam Abdelgawad. Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): Methodology and large-scale application on downtown toronto. *Intelligent Transportation Systems, IEEE Transactions on*, 14(3):1140–1150, 2013. ISSN 1524-9050. doi: 10.1109/TITS.2013.2255286.

Matthew Grounds and Daniel Kudenko. Parallel reinforcement learning with linear function approximation. In Karl Tuyls, Ann Nowe, Zahia Guessoum, and Daniel Kudenko, editors, *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*, volume 4865 of *Lecture Notes in Computer Science*, pages 60–74. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-77947-6.

Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138, December 2012.

R. Matthew Kretchmar. Parallel reinforcement learning. In *The 6th World Conference on Systemics, Cybernetics,and Informatics*, 2002.

Masayuki Kushida, Kenichi Takahashi, Hiroaki Ueda, and Tetsuhiro Miyahara. A comparative study of parallel reinforcement learning methods with a pc cluster system. In *Intelligent Agent Technology, 2006. IAT '06. IEEE/WIC/ACM International Conference on*, pages 416–419, Dec 2006. doi: 10.1109/IAT.2006.3.

Patrick Mannion, Jim Duggan, and Enda Howley. An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In Apostolos Kotsialos, Franziska Kluegl, Lee McCluskey, Joerg P Mueller, Omer Rana, and Rene Schumann, editors, *Autonomic Road Transport Support Systems*, Autonomic Systems. Birkhauser/Springer, 2015a. In Press.

Patrick Mannion, Jim Duggan, and Enda Howley. Parallel reinforcement learning for traffic signal control. In *Proceedings of the 4th International Workshop on Agent-based Mobility, Traffic and Transportation Models, Methodologies and Applications (ABMTRANS 2015)*, June 2015b. In Press.

Patrick Mannion, Jim Duggan, and Enda Howley. Parallel learning using heterogeneous agents. In *Proceedings of the Adaptive and Learning Agents Workshop (at AAMAS 2015)*, May 2015c.

Patrick Mannion, Jim Duggan, and Enda Howley. Learning traffic signal control with advice. In *Proceedings of the Adaptive and Learning Agents Workshop (at AAMAS 2015)*, May 2015d.

Christopher J.C.H. Watkins and Peter Dayan. Technical note: Q-learning. *Machine Learning*, 8(3-4):279–292, 1992. ISSN 0885-6125. doi: 10.1023/A:1022676722315.

Marco Wiering and Martijn van Otterlo, editors. *Reinforcement Learning: State-of-the-Art*. Springer, 2012.