Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 52 (2015) 956 – 961

Procedia Computer Science

The 4th International Workshop on Agent-based Mobility, Traffic and Transportation Models, Methodologies and Applications (ABMTRANS)

# Parallel Reinforcement Learning for Traffic Signal Control

Patrick Mannion[a,*], Jim Duggan[a], Enda Howley[a]

[a]*Discipline of Information Technology, National University of Ireland, Galway*

**Abstract**

Developing Adaptive Traffic Signal Control strategies for efficient urban traffic management is a challenging problem, which is not easily solved. Reinforcement Learning (RL) has been shown to be a promising approach when applied to traffic signal control (TSC) problems. When using RL agents for TSC, difficulties may arise with respect to convergence times and performance. This is especially pronounced on complex intersections with many different phases, due to the increased size of the state action space. Parallel Learning is an emerging technique in RL literature, which allows several learning agents to pool their experiences while learning concurrently on the same problem. Here we present an extension to a leading published work on RL for TSC, which leverages the benefits of Parallel Learning to increase exploration and reduce delay times and queue lengths.



Peer-review under responsibility of the Conference Program Chairs

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

## 1. Introduction

Traffic congestion is a major issue in modern cities that results in many negative environmental, social and economic consequences. High vehicle usage rates, coupled with the lack of space and public funds available to construct new transport infrastructure, serve to further complicate the issue. Against this backdrop, it is necessary to develop intelligent and economical solutions to improve the quality of service for road users. A relatively inexpensive way to alleviate the problem is to ensure optimal use of the existing road network, e.g. by using Adaptive Traffic Signal Control (ATSC). Improvements in ATSC have a pivotal role to play in the future development of Smart Cities, especially considering the current EU-wide emphasis on the theme of Smart, Green and Integrated Transport in Horizon 2020. Developing ATSC strategies for efficient urban traffic management is a challenging problem, which is not easily solved.

In recent years, Artificial Intelligence methods such as Fuzzy Logic, Neural Networks, Genetic Algorithms and Reinforcement Learning have all been applied successfully to the traffic control problem. This coincides with an increasing interest among researchers in the broader field of Intelligent Transportation Systems (ITS). The approach that we present in this paper is based on Reinforcement Learning (RL), a technique that has many potential applications in

* Corresponding author. Tel.: +353-851571492

*E-mail address:* p.mannion3@nuigalway.ie



Peer-review under responsibility of the Conference Program Chairs

doi:10.1016/j.procs.2015.05.172

the ITS area. Here we present an extension to a leading published work on Reinforcement Learning for Traffic Signal Control (RL-TSC), which leverages the benefits of Parallel Learning to increase exploration and reduce delay times and queue lengths.

This paper will make a contribution in the following ways: 1) a new method of Parallel Reinforcement Learning for Traffic Signal Control applications is presented; 2) we prove experimentally that the proposed method improves the quality of the solution reached compared to a standard single agent approach; 3) we identify and discuss specific issues that need to be taken into account when applying Parallel Learning to this difficult problem domain; 4) we discuss the future direction of this research topic.

The remainder of this paper is structured as follows: the second section discusses related research, while the third section describes our Parallel Learning approach. The following section details the design of our experimental set up, after which we present our experimental results. Finally, we conclude by discussing our findings and our plans for future work.

## 2. Related Research

### 2.1. Reinforcement Learning

The term Reinforcement Learning describes a class of algorithms that have the capability to learn through experience. An RL agent is deployed into an environment, usually without any prior knowledge of how to behave. The agent interacts with its environment, and receives a scalar reward signal $r$ based on the outcomes of previously selected actions. This reward can be either negative or positive, and a properly designed reward function allows the agent to iteratively learn an optimal or near optimal control policy. The agent must strike a balance between exploiting known good actions and exploring the consequences of new actions in order to maximise the reward received during its lifetime. Q values represent the expected reward for each state action pair, which aid the agent in deciding which action is most desirable to select when in a certain state. The Q values are typically stored in a matrix, which represents the knowledge learned by an RL agent.

RL problems are generally modelled as a Markov Decision Process (MDP), which is considered the de facto standard when formalising problems involving learning sequential decision making [1]. An MDP may be represented using a reward function $R$, set of states $S$, set of actions $A$, and a transition function $T$ [1], i.e. a tuple $< S, A, T, R >$. When in any state $s \in S$, selecting an action $a \in A$ will result in the environment entering 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 may be categorised as either model-based (e.g. Dyna, Prioritised Sweeping), or model-free (e.g. Q-Learning, SARSA). To successfully implement model-based approaches, it is necessary to know the transition function $T$ [1]. This can sometimes be problematic, given that $T$ may be difficult or even impossible to determine in complex problem domains. By contrast, model-free approaches do not have this requirement. Instead, model-free approaches sample the underlying MDP in order to gain knowledge about the unknown model. Exploration is necessary for a model-free learner in order to gain the required knowledge about its environment.

Q-Learning is an off-policy, model-free learning algorithm that is commonly used in RL-TSC literature, e.g. [2]. It has been proven that Q-learning converges 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 [3]. In Q-Learning, the Q values are updated according to the following equation:

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

The learning rate $\alpha \in [0, 1]$ is an input parameter required for many RL algorithms, and determines by how much Q values are updated at each time step $t$. The discount factor $\gamma \in [0, 1]$ controls how the agent regards future rewards.

### 2.2. Parallel Reinforcement Learning

Parallel Learning (PL) is an emerging paradigm within RL where multiple agents pool their experiences while learning concurrently on a problem, thus improving performance and decreasing convergence times. In contrast to mainstream Multi Agent Reinforcement Learning (MARL) research, PL is typically applied to speeding up the

convergence of single agent RL problems, rather than examining competitive or cooperative strategies and emergent behaviour as in MARL. PL agents typically influence each other's behaviour by sharing information, rather than the direct agent interactions that typically occur in MARL. This is because PL agents learn in separate instances of the same problem, rather than multiple agents learning in a single problem instance as is common in MARL literature.

Although several authors have reported promising results using Parallel Reinforcement Learning (PRL) approaches (see e.g.[4,5,6,7]), surprisingly little research has been published in this area to date. The main focus so far has been on testing PRL approaches using abstract problem domains like Gridworld, with the exception of Barrett et al.[4], who also tested their algorithm on a realistic cloud resource allocation problem. In general, previously published works in PL have reported improvements in convergence times and/or quality of the solution reached. These results are derived mainly from experimentation using simple abstract problem domains, but we hypothesise that PL techniques may also be effective when applied to more complex Traffic Signal Control problems.

## 3. Parallel Reinforcement Learning for Traffic Signal Control

Traffic Signal Control is a complex and highly stochastic problem domain, which presents a number of significant challenges when compared with the traditional abstract problem domains (e.g. Gridworld) studied in RL research. Specifically, the complexity of transportation networks and the number of independent entities involved is on such a scale that a learning agent cannot possibly know every detail about the environment state, therefore making these problems into Partially Observable Markov Decision Processes.

In RL-TSC, each intersection is typically controlled by a single agent. Each agent has the responsibility of determining the light switching sequence at its assigned intersection. A network of traffic signal control agents may then be considered as a Multi Agent System. Reinforcement Learning has been shown to be a promising approach when applied to urban traffic signal control (e.g.[2,8,9]), and offers many benefits; RL agents can learn online to continuously improve their performance, as well as adapting readily to changes in traffic demand. Traffic control problems make very attractive testbeds for emerging RL approaches, and present a number of non-trivial challenges such as developing strategies for coordination and information sharing between individual agents. For a comprehensive review of the usage of learning agents in Traffic Signal Control, we refer the interested reader to a review paper published by Mannion et al.[8].

A significant challenge of applying RL to traffic signal control relates to the number of possible state action combinations for complex intersections with many phases. This problem is referred to as the Curse of Dimensionality in RL literature. As RL agents encounter more complex problems, convergence times and the quality of the policy learned tend to suffer. Here we apply the principles of PRL to develop a method which can speed up learning and reach a better final policy compared to conventional RL-TSC approaches.

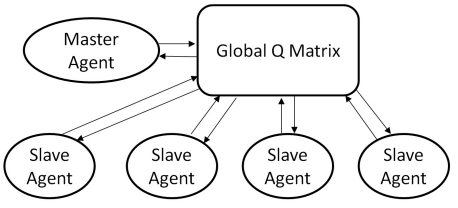


Fig. 1: Agents share their experience via a global Q matrix

Our PRL implementation consists of two types of agents: Master, and Slave. Both types of agents are based on Q-Learning, and 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 may 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. Here we use the same state, action and reward definitions as used by El-Tantawy et al.[2], so our PRL method can be considered to be an extension of this approach.

For each agent, state is defined as a vector of dimension $2 + P$, shown in Equation 2 below. The first 2 components are the index of the current phase ($P_c$) and the elapsed time in the current phase ($PTE$). The remaining $P$ components are the queue lengths ($QL_i$) for each phase. For a given state $s$, the state vector is defined as follows:

$$s = \left[ P_c, PTE, QL_1, ..., QL_n \right] \tag{2}$$

We limit the maximum number of queueing vehicles considered by an agent to 20, and the maximum phase elapsed time considered is limited to 30 seconds. By imposing these limits, we reduce the possible number of states considered by an agent. A vehicle is considered to be queueing at a junction if its approach speed is less than 10 km/hr.

The actions available to the agents at each time step are: to keep the currently displayed green and red signals, or to set a green light for a different phase. Phases are subject to a minimum length of 10 seconds, to eliminate unreasonably low phase lengths from consideration. There is no fixed cycle length, and agents are free to extend the current phase or switch to the next phase as they see fit. When changing 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.

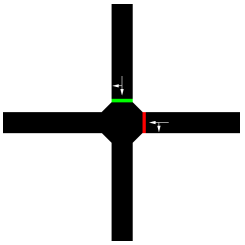
Actions are selected using the $\epsilon$-greedy strategy, where a random action is chosen with probability $\epsilon$, or the action with the best expected reward is chosen with the remaining probability $1-\epsilon$. The value of $\epsilon$ is set to 0.05. The learning rate $\alpha$ is set to 0.08, while the discount factor $\gamma$ is set to 0.8. These values for $\epsilon$, $\alpha$ and $\gamma$ are used by all learning agents in our experiments. All agents begin with their Q values for each state action pair initialised to zero at the start of each experiment.

The reward received by an agent for selecting an action $a$ in a given state $s$ and transitioning to a resultant state $s'$ is defined as the difference between the current and previous cumulative waiting times (CWT) of vehicles queueing at the junction. This reward function is shown in Equation 3 below. An action that results in an increase in CWT receives a negative reward, while an action that results in a decrease in CWT results in a positive reward.

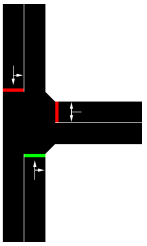
$$R(s, a, s') = CWT_s - CWT_{s'} \tag{3}$$

## 4. Experimental Design

Fig. 2: Test Junctions



(a) 2 Phase Junction



(b) 3 Phase Junction

The microscopic traffic simulation package SUMO (Simulation of Urban MObility) [10] is the basis for our experimental setup. Agent logic is defined in our external framework, which is implemented in Java. The simulation timestep length is 1 second for all experiments. Each RL agent is responsible for controlling the light sequences of a single junction. The TraaS library [11] is used to feed simulation data to the agents, and also to send commands from the agents back to SUMO.

The hourly traffic demand $D$ used in our experiments is defined as a step function. The demand step function comprises a base flow $b$, episode length $e$, step demand increase $h$, and a step interval $i$. Demand at time $t$ into a particular episode is defined as shown in 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 is a more realistic and demanding flow pattern for the agents to control, compared to a constant hourly demand definition. These step functions aim to simulate a peak in demand similar to a morning or evening rush hour. The episode length $e$ used for all intersections is two hours. Each experiment is run for 75 successive two hour episodes, with the same demand step function repeated over each two hour episode.

The first experimental scenario is a simplified junction with 2 phases: North and East (see Fig. 2a). Here two intersecting one-way streets are controlled by a set of traffic lights, with the number of phases $P = 2$. The base flow for the step function is 1000 vehicles per hour (veh/hr). The demand level rises by 250 veh/hr every 15 minutes, reaching a peak of 1750 veh/hr, before stepping down again to a value of 1000 veh/hr. There are four possible routes through the intersection, to which vehicles are randomly assigned upon creation in the simulator.

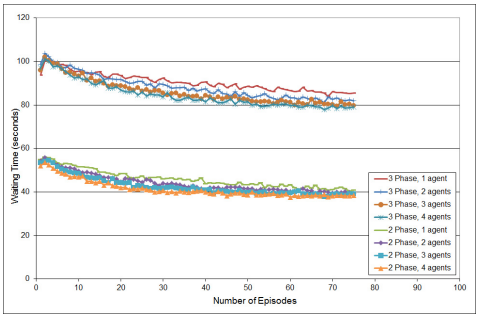
The second test scenario consists of three intersecting two-way streets, which form a T junction (shown in Fig. 2b). Here there are 6 possible routes through the junction arising from the lane configuration, and three phases: North, East and South. The base flow is 1000 veh/hr, rising by 100 veh/hr every 15 minutes to a peak of 1300 veh/hr, before returning to 1000 veh/hr.

For each experimental scenario, we first test a single RL agent, based on those used by El-Tantawy el al.[2]. This approach has already been proven to offer performance improvements compared to real world traffic control systems based on fixed-time control, semiactuated control, and SCOOT control.

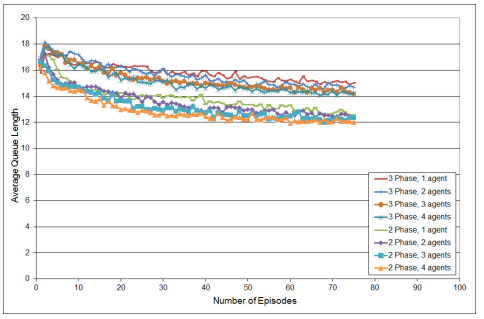
We then test our parallel learning algorithm against this, with 2, 3 and 4 agents learning simultaneously on the same problem. As other works have already proved the efficacy of RL-TSC approaches in test scenarios with multiple intersections (see e.g.[2,8]), we have decided to focus on using single junction test cases to clearly illustrate the benefit of our PRL approach without adding unnecessary additional complexity.

## 5. Experimental Results and Discussion

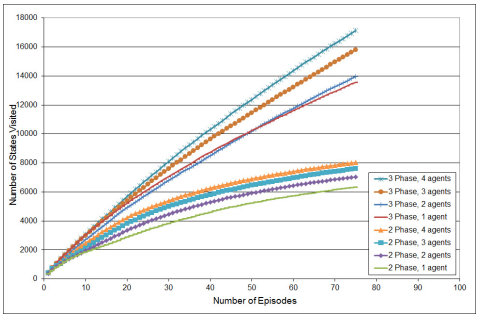
Fig. 3: Experimental Results




(a) Average Waiting Times




(b) Average Queue Lengths




(c) Average Number of States Visited

We evaluate each approach tested using the following parameters: Average Waiting Times (AWT), Average Queue Lengths (AQL), and Average Number of States Visited (ANSV). The values reported for AWT and AQL are the average value for each two hour episode, while for ANSV the actual value at the end of each two hour episode is reported. The results presented are the average of 10 runs, and are summarised in Table 1.

Fig. 3a shows the average vehicle waiting times (AWT) for the junctions tested. This plot shows an improvement in AWT compared to a single agent when multiple agents are applied to the same problem. The Average Queue Lengths (AQL) for each experiment are plotted in Fig. 3b. Similar to our findings in terms of AWT, our PRL approach beats a single learner on all the intersections tested, resulting in lower AQL values. Adding additional agents on both the 2 and 3 Phase junctions resulted in a further decrease in AWT and AQL, which scaled quite well relative to the number of additional learners.

Another benefit offered by our PRL approach is the increased exploration when compared with a single learning agent. Fig. 3c presents the Average Number of States Visited (ANSV) at the end of each episode for all tests. On both tested junctions, our PRL approach results in an increased number of states being reached when compared with a single agent. Multiple agents learning on the same problem results in increased diversity in terms of action selection choices, leading to knowledge being gained about a higher number of possible system states.

To ensure the significance of the results, a series of two-tailed t-tests were performed. Here we compared the average values of AWT, AQL, and ANSV over the final 10 episodes of each experiment for the single agent and PRL with 2 agents. The differences in the means were deemed to be significant if the p value was less than 0.05. PRL with 2 agents was found to offer a statistically significant improvement over the single agent approach in both test cases for all experimental parameters measured.

The observed improvement in performance also appears to scale well with the number of PRL agents applied to the problem. Reductions in AWT and AQL of the order of up to 7% and 5%

respectively are possible when using 4 PRL agents compared to a standard single learning agent approach. Increases of up to 27% recorded in ANSV show that PRL increases exploration when compared to a single learner.

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

| Experiment | AWT (s) | % Reduction AWT | AQL | % Reduction AQL | ANSV | % Increase ANSV |
|---|---|---|---|---|---|---|
| 2 Phase, 1 agent | 41.45 | - | 12.74 | - | 6,170.02 | - |
| 2 Phase, 2 agents | 39.84 | 3.88 % | 12.37 | 2.90 % | 6,874.23 | 11.41 % |
| 2 Phase, 3 agents | 38.66 | 6.73 % | 12.23 | 4.00 % | 7,425.38 | 20.35 % |
| 2 Phase, 4 agents | 38.29 | 7.62 % | 12.12 | 4.87 % | 7,868.37 | 27.53 % |
| 3 Phase, 1 agent | 85.52 | - | 15.06 | - | 13,001.19 | - |
| 3 Phase, 2 agents | 82.45 | 3.59 % | 14.77 | 1.93 % | 13,326.58 | 2.50 % |
| 3 Phase, 3 agents | 80.40 | 5.99 % | 14.42 | 4.25 % | 15,035.03 | 15.64 % |
| 3 Phase, 4 agents | 78.86 | 7.79 % | 14.21 | 5.64 % | 16,322.52 | 25.55 % |

## 6. Conclusions and Future Work

Here we have presented a novel method of Parallel Reinforcement Learning for Traffic Signal Control, and proven experimentally that it outperforms a single learning agent approach. Our PRL algorithm increased exploration, while also reducing waiting times and queue lengths when compared to a single RL agent learning on the same problem. By testing 2, 3 and 4 agents learning in parallel, we have also shown that additional performance gains are possible by increasing the number of parallel learners. In the future we plan to test this method more extensively on simulated real world traffic networks with multiple signalised junctions. We also wish to investigate the effect of using alternative exploration strategies with PRL. In previous work [7] we partitioned the state action space of a simple PRL problem, with a single PRL agent responsible for learning about each subset of the state action space, allowing for more focused exploration. Using this exploration strategy in conjunction with our PRL algorithm for TSC has produced promising initial results, and will allow for further improvements in performance beyond those presented in this paper.

## Acknowledgments

The primary author would like to acknowledge the financial support provided to him by the Irish Research Council.

## References

1. Wiering, M., van Otterlo, M., editors. *Reinforcement Learning: State-of-the-Art*. Springer; 2012.
2. 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. *Intelligent Transportation Systems, IEEE Transactions on* 2013;**14**(3):1140–1150. doi:10.1109/TITS.2013.2255286.
3. Watkins, C., Dayan, P.. Technical note: Q-learning. *Machine Learning* 1992;**8**(3-4):279–292. doi:10.1023/A:1022676722315.
4. Barrett, E., Duggan, J., Howley, E.. A parallel framework for bayesian reinforcement learning. *Connection Science* 2014;**26**(1):7–23.
5. Grounds, M., Kudenko, D.. Parallel reinforcement learning with linear function approximation. In: Tuyls, K., Nowe, A., Guessoum, Z., Kudenko, D., editors. *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning*; vol. 4865 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. ISBN 978-3-540-77947-6; 2008, p. 60–74.
6. Kretchmar, R.M.. Parallel reinforcement learning. In: *The 6th World Conference on Systemics, Cybernetics,and Informatics*. 2002, .
7. Mannion, P., Duggan, J., Howley, E.. Parallel learning using heterogeneous agents. In: *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2015)*. 2015 (in press), .
8. Mannion, P., Duggan, J., Howley, E.. An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In: Kotsialos, A., Kluegl, F., McCluskey, L., Mueller, J.P., Rana, O., Schumann, R., editors. *Autonomic Road Transport Support Systems*; Autonomic Systems. Birkhauser/Springer; 2015 (in press), .
9. Mannion, P., Duggan, J., Howley, E.. Learning traffic signal control with advice. In: *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2015)*. 2015 (in press), .
10. 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.
11. Traas: Traci as a service. 2015. URL: `http://traas.sourceforge.net/cms/`.