Reinforcement Learning for Adaptive Traffic Signal Control

Yun Chen
260772822
yun.chen4@mail.mcgill.ca

Wenting Wang
260367035
wenting.wang@mail.mcgill.ca

Yi Chang
260619034
yi.chang@mail.mcgill.ca

Abstract

Nowadays, traffic congestion has been very troublesome and wastes lots of time of urban residents whose quality of living has been reduced. Improvements in Adaptive Traffic Signal Control (ATSC) technology have helped build smart cities and relieve traffic congestion. The automatic typical method for ATSC is Reinforcement Learning (RL). In this project, we experimented applying Reinforcement Learning algorithms to the topic of adaptive traffic signal control in hope of reducing traffic queue length as well as waiting time. The attempted approaches in our project includes using both Q Learning and Dyna Q learning method with different Reward definition. Our experimentation demonstrated that adaptive traffic signal control using RL approach was able to reduce queue length and waiting time in a four-intersect cross road traffic architect.

1. Introduction

Urban cities are full of complex networks of roads and the traffic demands are quite high, especially in the rush hour. One way to serve the demands is to improve the infrastructure. However, it is not always possible due to space and financial restrictions. One significant approach is to reduce the travelling time based on the existing infrastructure through intelligent transportation system (ITS). Adaptive Traffic Signal Control (ATSC) has been a relatively economic way to ensure the optimal use of the existing road network [1].

There are many approaches that have been investigated including SCOOT [2] and SCATS [3], PRODYN [4], OPAC [5], UTOPIA [6], RHODES [7], etc. However, these models all require the exact model of environment that has been pre-constructed. Since the situation of traffic is usually stochastic, it would be simpler to control if the method can adapt to the changes of the environment. Reinforcement learning is such a way to self-learn from the experience and adapts. Besides, online reinforcement learning enables the real-time refinement of control policy and continuous self-optimization.

Problems of traffic control has been a very attractive test bed for RL algorithms and get non-trivial challenges such as developing strategies for coordination and information sharing between individual agents.

In the following parts, the three RL algorithms and the implementation details are shown. Besides, the discussion about the three algorithms and future work about this project are described.

2. Related Work

There has been significant research in the traffic signal control using RL methods. Generally speaking, traffic signals repeat *cycles*, which is a complete rotation through all of the indications provided at the intersection. Every cycle is made of a sequence of N phases. One *phase* is made of traffic flow movement through the intersection such as North to South and South to North. One typical phase has the time which is defined as *green* time followed by a duration of *yellow* and then *red*. The duration time is either a fixed plan by the generally used Webster method [8] or the adaptive green time of each phase and variable sequences of phases depending on traffic dynamics.

SARSA for traffic light control was introduced by Thorpe [9]. Abdulhai et al [10] introduced Q-learning for an isolated intersection. Bingham [11] introduced a neuro-fuzzy traffic signal controller that uses RL for learning the neural network. Besides, the RL method with context detection to solve control optimization problem is introduced by Olivera et al [12].

The most elementary applications of RL in traffic signal control only considers a single agent which is suitable for one isolated intersection but not feasible for larger networks such as the networks in small towns. Abdulhai et al [10] presents a case study of Q-learning for an isolated two-phase signal junction controlling. It is found that effect of Q-learning is the same as pre-timed signals on constant or constant-ratio flows and outperforms greatly under more variable flows.

Apart from single agent RL in traffic control, there are also multi-agent RL methods. One of the most significant early works in multi-agent RL method is that of Wier-

ing [13]. Several works have extended Wiering's approach [14, 15, 16, 17]. Wiering developed a mode-based approach and presented three algorithms and tested on a simple 3×2 grid network. Steingrver et al.[18] further extended the work of Wiering, by introducing a basic form of information sharing between agents, but all his approaches have the same problem, which is significant increase in the state space. In [19], Salkham and Cahill introduced a pattern change detection mechanism allowing the agent to re-learn based on the changes detected in traffic flows.

Multi-objectivity is an also a popular research theme in RL application in the traffic control area [20, 21, 22]. Most of the traditional traffic control methods are single objective [22], which seeks optimal solutions based on a single parameter only. Multi-objectivity typically is achieved by expanding the agent's reward function definition and including multiple parameters. Houli et al [23] presented a multi-objective algorithm. It is tuned for three different scenarios: light, medium, and heavy traffic and each level of traffic has a different Q function. A more advanced approach is introduced Khamis et al [20]. The author developed multi-objective with weighted sum reward functions of different agents. In our work, we applied the multi-objectivity by defining the reward function as the average of all agents' reward.

3. Approaches

In this section, we will first explain how we transformed the problem of traffic signal control into a environment that RL agents can learn from. This includes the definition of state space, action space and reward. Then, we will cover RL learning mythologies that used to learn from the predefined environment.

3.1. Environment

An OpenAI, Gym environment [24] is defined with the help from the following two open source packages to simulate the learning environment.

- Simulation of Urban Mobility, SUMO [25] for microscopic traffic simulation
- TraCi [26] for communication between agents and SUMO in Python

Definition of state, action and reward of the environment will be discussed in detail in the following subsections.

3.1.1 State Definition

As introduced in paper "Adaptive Traffic Signal Control: Exploring Reward Definition For Reinforcement Learning" written by Saad Touhbi [27], we represent the state in our environment by $2+P$ parameters. The first two parameters keeps track of the status of the current phase a traffic light is set on at a given time. The two parameters are the index of the current phase and elapsed time of the current phase respectively. The P components are the maximum queue length s_i^t for each phase at a given time t. This $2+P$ parameters for a state is then treated as radix mixed representation, and transformed into a single number. This number is finally defined as state in our environment.

For the definition of queue length, Touhbi demonstrated a form that takes both the lane length and vehicle's dimension into consideration [27]. Equation 1 shows the definition of the queue length by Touhbi, where q_k^t is the queue length in meters on lane k in time t and l_k is the length of lane k. Then define lq_k^t as the queuing level which is the queue length on lane k at time t divided by length of the lane [27] as the following equation shows:

$$lq_k^t = \frac{q_k^t}{l_k} \tag{1}$$

Furthermore, assume the number of phases is P in one cycle. Given L_i is the set of active lanes in phase i, the maximum queuing in each phase i at time t is defined as in equation 2 [27].

$$s_i^t = \max_{k \in L_i} lq_k^t \tag{2}$$

3.1.2 Action Definition

Similar to Touhbi's approach [27], the agent has P actions to take at each decision making time to set the traffic light to any of the possible P phases. However, every time a traffic light switch phase, it has to go through a yellow phase on the current green light for $yellow$ seconds then followed by a all red phase for $allred$ seconds to ensure safety. Furthermore, each phase will have a minimum of $mingreen$ seconds of being active to avoid hazardously frequent light switching.

With the above defined safety constraints, if a new active phase is selected at time t, then the next iteration of decision making for the agent will happen at time $t + yellow + allred + mingreen + 1$. If the new phase matches the current phase, then the next iteration will happen at time $t+1$ second. It means that if a green light is on, the lighting time of green must not be longer than a minimum green time, $mingreen$. After the minimum time, the state of system will be evaluated in every second.

As Toubhi explains [27], let $a_t = i, i \in 1, ..., N$ denotes the action taken at time t and i is the number of action taken. If an iteration of decision making happens at time t, $p(t)$ then denotes the previous iteration of decision making. If the current action equals the previous action, which is denoted as $a_t = a_{p(t)}$, the result of current action will be to extend the green light by 1 second. After one more second, the decision making process is launched. However, if $a_t \neq a_{p(t)}$, after accounting for the yellow and all red on current phase,

the phase will be changed. The minimum green on the chosen next phase will last and one new iteration of decision making starts after a time of *mingreen*.

3.1.3 Reward Definition

In our project, we have experimented with two reward definitions. The first one is defined as the difference between the sum of squared maximum queue length between two decision making time, precisely as shown in Equation 3 [27]. In our report, we refer this reward definition as 'RLTSC-1' for the ease of understanding. In this definition, a positive reward will be given if the queue length before action taken is longer than queue length action taken. Otherwise, if the action taken results a longer queue length, a negative reward is given.

$$R_t = \sum_{i \in N} (\max_{l \in L_i} q_l^{p(t)})^2 - \sum_{i \in N} (\max_{l \in L_i} q_l^t)^2 \tag{3}$$

For the second definition of reward, we used the similar definition as above, but with a multi-objective approach. a reward for a agent's action is not only depends on queue length on the incoming lanes for the single junction, but it takes consideration of queue length changes on all lanes. In our report, we refer this reward definition as 'RLTSC-2' for the ease of understanding

In our project, both reward definitions are used in each learning method for finding the optimal approach.

3.2. Learning Methods

For learning methods, we have applied two approaches, including Q learning and Dyna Q learning. We will these two approaches in details in the following subsections.

3.2.1 Q-learning

Q-learning is an off-policy, model-free algorithm. The Q-learning values are updated by the equation below:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha(r_t + \gamma Q_{max_a}(s_{t+1}, a) - Q_t(s_t, a_t)) \tag{4}$$

where r_t is the reward at iteration time t when performing action a on a state. α is the learning rate meaning the level of dependence between past knowledge and the new knowledge. γ is the discount number meaning the importance of next state. α and γ both are parameters between 0 and 1. If the algorithm has a higher learning rate, the agent will learn more on the knowledge acquired in the current action. If the discount factor is closer to 1 and it means that the resulting state is important and has influence on performance of the agent. The learning rate is

$$\alpha_j = \frac{1}{ln(v_j(s,a))} \tag{5}$$

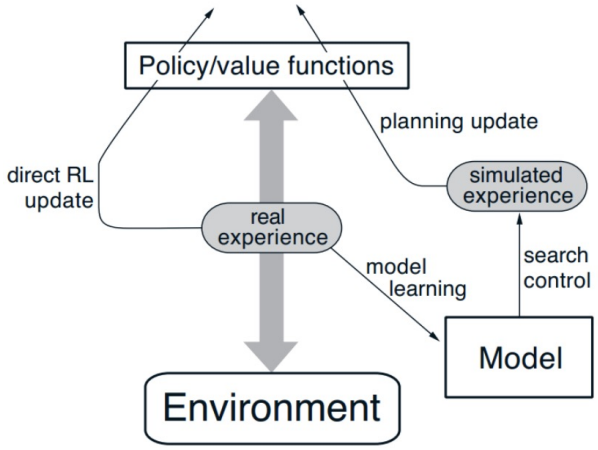

Figure 1. General structure of Dyna-Q algorithm.

where $v_j(s,a)$ is the number of visits to a particular state-action pair. It has the meaning that firstly, the learning rate is high but reduces towards to 0 after enough exploration. With time going on, the importance of future experiences on Q-value decreases. The discount factor we choose is constant as 0.05.

For selection of action on each iteration of decision making, we applied the $\varepsilon - greedy$ method, which means the best option is chosen with probability ε and a random action is chosen with probability $1-\varepsilon$. In our project, we have chosen $\varepsilon = 0.9$

3.2.2 Dyna-Q Learning

Dyna-Q is a typical learning algorithm which combines Q-learning and Q-planning. The planning is one-step tabular Q-planning and the learning is one-step Q-learning. The real or simulated experience increase the effect of model through learning model and value function and policy. The model contains state, action, next state and reward tuples.

Figure 1 shows the general structure of Dyna-Q learning algorithm. The real experience passing back and forth between the environment and policy affects policy and value functions. The level of effectiveness equals to the simulated experience generated by model of environment.

Similiar to Q learning method, for selection of action on each iteration of decision making, we applied the $\varepsilon - greedy$ method with $\varepsilon = 0.9$

4. Experiment Implementation

An OpenAI, Gym environment [24] is defined with the help from the following two open source packages introuded in the previous section to simulate the learning environment.

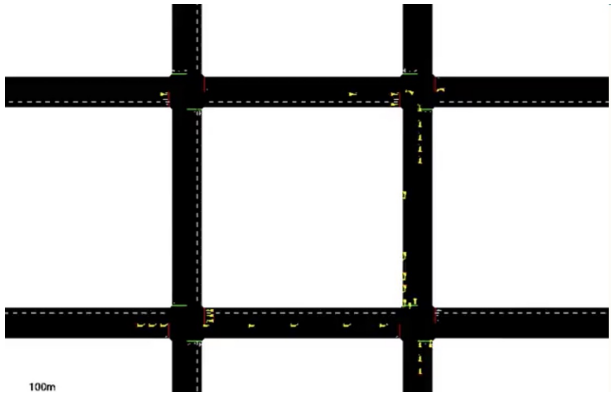

Figure 2. A screen shot of simulated four junctions of traffic.

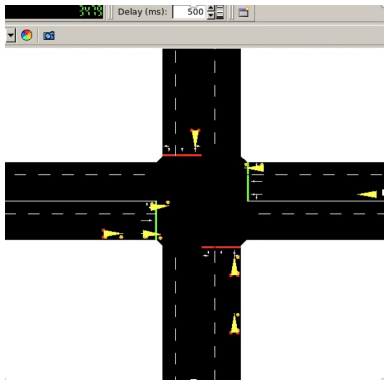

Figure 3. A screen shot of simulated one intersection where the lights are green in the east and west direction.

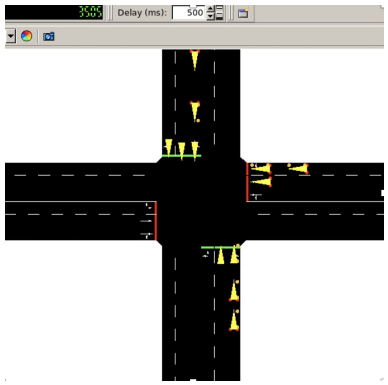

Figure 4. A screen shot of simulated one intersection where the lights are green in the north and south direction.

For the sake of computational feasibility, we restrict the number of phases to be 2, the maximum elapsed time to be 30 seconds and maximum queue length to be 20. These numbers are set to keep the number of states with in $2 \times 30 \times 20 \times 20 = 24000$, so the problem is computationally achievable in our project. A more realistic environment that involves a larger number of states is consider as future work to our project.

The actions for agents at every time step are to keep the current green and red signals or to set a new green light for a different phase. The minimum time for a phase is set to be 20 seconds to ensure that there is no unreasonable low-length phase time. Since the length of a phase is not fixed, the agent can change arbitrarily from one phase to another phase as long as they fit the situation. At every phase-switch, an yellow signal will need to be set for 3s and followed by a all red period for 2s. The light is finally switched to the next phase and set for at least 20 seconds. The 3 steps described above need to be completed for each phase-switch without any interruption.

We test the experiment on the 2×2 grid network. As a result, we have 4 junctions. Figure 2 is a screen shot of our simulated four intersections. Figure 3 is one screen shot of intersection. There are 4 agents used in controlling traffic signals in our network. Each agent is responsible for controlling only traffic light at a single junction. We evaluate both Q-Learning and Dyna-Q learning algorithms using both RLTSC-1 and RLTSC-2 reward definitions.

Figure 2 is a screen shot of our simulated four intersections. Figure 3 is one screen shot of intersection.

5. Results and Conclusion

Figure 3 and figure 4 are both screen shots of the same intersection. We can see that from Figure 3 to 4, the time shown in the above of figures have lasted for about 25s. By default, the time between two different phases is 30s. As a result, the time reduced than the default value showing that our algorithm have learned and improved traffic congestion.

For evaluation, random traffic flow are generated using DUAROUTER [28] tool to simulate live traffic environment.Figure 5 illustrates that that our experimented approaches have achieved reducing both queue length and waiting time in traffic when comparing with using fixed traffic signal control. For each of the method, average queue length and waiting time for our entire traffic network shown in Figure 5 is calculated over 10000 time steps and graphed as a bar diagram.

As shown above, the approach which performs best is Q Learning method with RLTSC-2 reward definition. This is expected, since Q Learning is more suitable for a stochastic situation, such as random traffic. Dyna Q is less performant in this environment, as it has planning involved. Between the two reward definitions, RLTSC-2 gives more promising result, since it does not only consider queue length of its own, but as well as queue length at the other junctions. It is a better performant reward definiton when we are targetting to enhance traffic flow of the entire network.

After all, the most performant approach shows its capa-

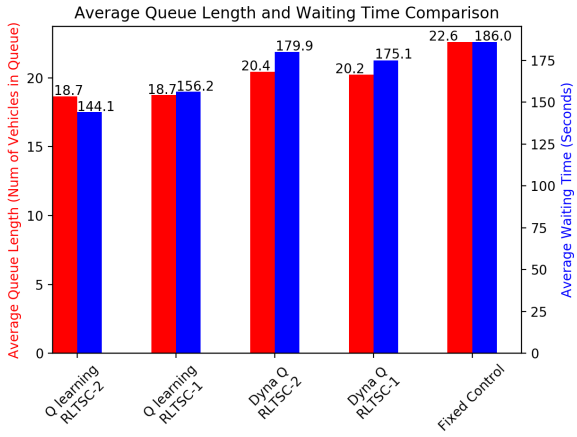

Figure 5. Average Queue Length and Waiting Time Comparison

bility of reducing queue length level as well as waiting time to more than 40 seconds at every time step.

6. Discussion and Future Work

There are several directions of the future work. Firstly, when the number of conjunctions of road network is large, the correlations between different junctions cannot be ignored. In this siuations, we could try some multi-agent algorithms. Secondly, besides the waiting time and queue length considered in the reward definition, we could consider other attributes in the reward, like the CO2 emission. Thirdly, when simulating the real-world traffic with SUMO, we don't set the signal for left turn only or right turn only. we could try to set these lights and take them considered in the algorithms. In this way, we could better simulate the real-word traffic and build the corresponding model. Finally, we could try some advanced RL algorithms, like neural network.

7. Contribution

Yi Chang did the research, selected the topic, defined the state and reward for RL algorithm, and also set up the simulation environment SUMO. Wenting Wang set up the environment and implement the algorithms, Yi Chang also worked on the DynaQ's implementation. Weniting Wang also made the presentation. Comprehensive report writing is done by everyone, but Yun Chen makes the biggest contribution. Yun Chen also prepared the slides for presentation.

References

[1] Dennis I Robertson. Transyt: a traffic network study tool. 1969.

[2] PB Hunt, DI Robertson, RD Bretherton, and RI Winton. Scoot-a traffic responsive method of coordinating signals. Technical report, 1981.

[3] Arthur G Sims and Kenneth W Dobinson. The sydney coordinated adaptive traffic (scat) system philosophy and benefits. *IEEE Transactions on vehicular technology*, 29(2):130–137, 1980.

[4] Jean-Jacques Henry, Jean Loup Farges, and J Tuffal. The prodyn real time traffic algorithm. In *Control in Transportation Systems*, pages 305–310. Elsevier, 1984.

[5] Nathan H Gartner. *OPAC: A demand-responsive strategy for traffic signal control*. Number 906. 1983.

[6] Vito Mauro and C Di Taranto. Utopia. In *Control, computers, communications in transportation*, pages 245–252. Elsevier, 1990.

[7] F Donati, Vito Mauro, G Roncolini, and M Vallauri. A hierarchical-decentralized traffic light control system. the first realization:progetto torino. *IFAC Proceedings Volumes*, 17(2):2853–2858, 1984.

[8] FV Webster. Traffic signal settings, road research technical paper no. 39. *Road Research Laboratory*, 1958.

[9] Thomas L Thorpe. Vehicle traffic light control using sarsa. In *Online]. Available: citeseer. ist. psu. edu/thorpe97vehicle. html*. Citeseer, 1997.

[10] Baher Abdulhai and Lina Kattan. Reinforcement learning: Introduction to theory and potential for transport applications. *Canadian Journal of Civil Engineering*, 30(6):981–991, 2003.

[11] Ella Bingham. Reinforcement learning in neurofuzzy traffic signal control. *European Journal of Operational Research*, 131(2):232–241, 2001.

[12] Denise de Oliveira, Ana LC Bazzan, Bruno Castro da Silva, Eduardo W Basso, Luis Nunes, Rosaldo Rossetti, Eugénio de Oliveira, Roberto da Silva, and Luis Lamb. Reinforcement learning based control of traffic lights in non-stationary environments: A case study in a microscopic simulator. In *EUMAS*, 2006.

[13] Marco Wiering. Multi-agent reinforcement leraning for traffic light control. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 1151–1158, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[14] B Bakker, M Steingrover, R Schouten, Emil Nijhuis, and L.J.H.M. Kester. Cooperative multi-agent reinforcement learning of traffic lights. 01 2005.

[15] Bram Bakker, Shimon Whiteson, Leon Kester, and Frans C. A. Groen. *Traffic Light Control by Multi-agent Reinforcement Learning Systems*, pages 475–510. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[16] Ji I*vs*.a, J. J. Sandra Kooij, Rogier Koppejan, and Lior Kuijer. Reinforcement learning of traffic light controllers adapting to accidents. 2006.

[17] M. Wiering, J. Vreeken, J. van Veenen, and A. Koopman. Simulation and optimization of traffic in a city. In *IEEE Intelligent Vehicles Symposium, 2004*, pages 453–458, June 2004.

[18] Merlijn Steingrver, Roelant Schouten, Stefan Peelen, Emil Nijhuis, and Bram Bakker. Reinforcement learning of traffic light controllers adapting to traffic congestion. In *In Proceedings of the Belgium-Netherlands Artificial Intelligence Conference, BNAIC05*, 2005.

[19] A. Salkham and V. Cahill. Soilse: A decentralized approach to optimization of fluctuating urban traffic using reinforcement learning. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 531–538, Sept 2010.

[20] M. A. Khamis and W. Gomaa. Enhanced multiagent multi-objective reinforcement learning for urban traffic light control. In *2012 11th International Conference on Machine Learning and Applications*, volume 1, pages 586–591, Dec 2012.

[21] Mohamed A. Khamis and Walid Gomaa. Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework. *Engineering Applications of Artificial Intelligence*, 29:134 – 151, 2014.

[22] M. A. Khamis, W. Gomaa, and H. El-Shishiny. Multi-objective traffic light control system based on bayesian probability interpretation. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 995–1000, Sept 2012.

[23] Duan Houli, Li Zhiheng, and Zhang Yi. Multiobjective reinforcement learning for traffic signal control using vehicular ad hoc network. *EURASIP J. Adv. Signal Process*, 2010:7:1–7:7, March 2010.

[24] OpenAI. `https://gym.openai.com/envs/#classic_control`.

[25] Simulation of Urban Mobility. `http://sumo.dlr.de/wiki/TraCI`.

[26] TraCI. `http://sumo.dlr.de/wiki/TraCI`.

[27] Patrick Mannion, Jim Duggan, and Enda Howley. Parallel reinforcement learning for traffic signal control. *Procedia Computer Science*, 52:956–961, 2015.

[28] DUAROUTER. `http://sumo.dlr.de/wiki/DUAROUTER`.