IET Intelligent Transport Systems

Special Issue: Selected Papers from the 25th ITS World Congress Copenhagen 2018

Value-based deep reinforcement learning for adaptive isolated intersection signal control

ISSN 1751-956X
Received on 26th January 2018
Revised 26th April 2018
Accepted on 8th June 2018
E-First on 20th August 2018
doi: 10.1049/iet-its.2018.5170
www.ietdl.org

Chia-Hao Wan[1] ✉, *Ming-Chorng Hwang*[1]

[1]*ITS Research Center, China Engineering Consultants, Inc., 28F, No.185, Sec.2, Sinhai Rd., Taipei City, Taiwan*

✉ *E-mail: tenthousand@ceci.org.tw*

Abstract: Under efficiency improvement of road networks by utilizing advanced traffic signal control methods, intelligent transportation systems intend to characterize a smart city. Recently, due to significant progress in artificial intelligence, machine learning-based framework of adaptive traffic signal control has been highly concentrated. In particular, deep Q-learning neural network is a model-free technique and can be applied to optimal action selection problems. However, setting variable green time is a key mechanism to reflect traffic fluctuations such that time steps need not be fixed intervals in reinforcement learning framework. In this study, the authors proposed a dynamic discount factor embedded in the iterative Bellman equation to prevent from a biased estimation of action-value function due to the effects of inconstant time step interval. Moreover, action is added to the input layer of the neural network in the training process, and the output layer is the estimated action-value for the denoted action. Then, the trained neural network can be used to generate action that leads to an optimal estimated value within a finite set as the agents' policy. The preliminary results show that the trained agent outperforms a fixed timing plan in all testing cases with reducing system total delay by 20%..

1 Introduction

Adaptive signal control has always been regarded as advanced traffic control technology. Traditional control methods can't deal with the variability and unpredictability of traffic demand, but on the other hand, adaptive control is designed to be used in all kinds of traffic scenarios. By receiving and processing data from sensors, adaptive signal control technologies can determine when and how long the lights should be green and update signal timing settings to optimise intersection performance.

Over a long period of time, traffic engineers design lots of clever algorithms implemented on adaptive control. In general cases, adaptive control algorithms are designed to fit local traffic conditions and specific problems. Therefore, adaptive control can usually get good performance improvements, but yet it also needs lots of human efforts on algorithm design. That is a perfect occasion for artificial intelligence (AI) to deal with.

One of the core techniques in AI is reinforcement learning (RL). RL is a machine learning paradigm which concerned with how software agents ought to take actions in an environment in order to maximise the cumulative reward in a trajectory. In this study, we defined the intersection as our environment, the signal phase selection as the action and the intersection performance metrics (in here we adopted system total delay) as the reward. That is to say, the agent is seeking a signal phase selection strategy that will earn the highest performance in this intersection.

Another core technique in AI is so called deep learning (DL). DL is also one part of machine learning methods and it is often used to in describing the relation between input data and output data. In this case, the deep neural network can address traffic behaviour and transform it into a mathematical model. Basically, a deep neural network will combine all kinds of traffic information into a strategy value, and then the response to the RL process.

In this paper, we adopted a value-based deep reinforcement learning model which is modified from the deep Q-learning network (DQN) algorithm to implement on adaptive signal control. The agent training process was done in the microsimulator VISSIM, at an isolated intersection.

The remaining of this paper is organised as follows. The literature review is given in Section 2. Value-based models and detail settings are described in Sections 3. Scenario testing and result are presented in Sections 4. A brief summary and suggestion is discussed in the last part of this paper.

2 Literature review

RL is an area of machine learning and mainly deals with the problem which has a sequential decision-making process [1]. A classical RL model [2] can be expressed in Fig. 1.

In this model, the decision-maker is called an agent. The agent will interact with the environment via different actions. After an action, the state of the environment will be changed, and the environment will respond a reward index to the agent simultaneously. The goal of the agent is to maximise the cumulative rewards. The whole RL algorithm is to help the agent learn how to achieve that goal via sequential decision.

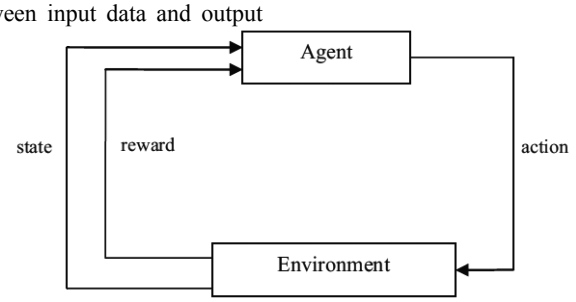

Fig. 1 *Classical RL framework*

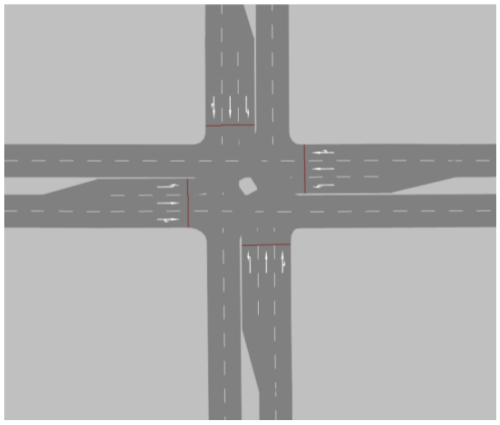

Fig. 2 *Isolated signalised intersection layout*

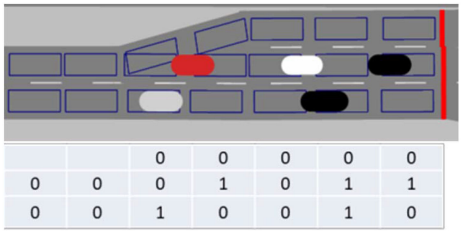

Fig. 3 *Using cell occupancy to describe traffic state*

There are three major categories of RL methods: dynamic programming, Monte Carlo and temporal difference (TD) learning methods. Dynamic programming algorithms are a model-based method which means that they require a model of the Markov decision process [3, 4]. Monte Carlo methods are a model-free method and learn directly from experience. They do not need a complete probability distribution of all possible transitions which is required in dynamic programming methods [1, 4]. TD learning is a combination of Monte Carlo ideas and dynamic programming ideas [1, 5]. TD learning methods do not need the entire trajectory sample. They update state value from the part of trajectory [1].

Q-learning is one of the temporal difference methods [6, 7]. The core of it is to learn an action-value function, $Q(s, a)$, which ultimately gives the expected utility of taking a given action a in a given state s, and following an optimal policy thereafter. One of the variants of Q-learning is deep Q-networks, also known as DQN [8]. Recent advances in deep neural networks [9–11] that allow DQN using DNN to play the role of action-value function $Q(s, a)$. DQN has made a huge success on master Atari games.

On the other hand, traffic signal control also attempts to introduce AI to create an innovative solution. Beginning in the early 2000's, several types of research adopted RL technique on adaptive signal control [12–14]. After google DeepMind first proposed the term 'Deep Reinforcement Learning (DRL)' in 2013 [15], it triggered another wave of using DRL in signal control. The authors [16, 17] have established a clear DRL signal control framework in the isolated intersection. They mainly focus on how to describe a traffic state or the actions space. Meanwhile, other research [18] dedicated to the stacked autoencoders to improve performance.

In this paper, we followed these experiences but focused more on whether different traffic scenarios will affect the agent's learning process and its performance.

3 Signal control DRL model

In this section, we would first introduce how to fit the signal control problem into an RL model. In other words, we would make a clear definition of state element, action space and reward. Then, we would introduce the algorithm of the traditional DQN method. At last, we would improve the inefficient part of DQN, and propose a value based state-action model.

Table 1 Specific traffic parameters to describe a state

Name	Meaning	Unit
current phase	Which phase is in the current state.	one hot list
green duration	For each signal head, how many seconds has the light been in green already. Using 0 for red and amber.	second
red duration	For each signal head, how many seconds has the light been in red already. Using 0 for green and amber.	second
left-turn bay occupation	How many cars are in the left-turn bay.	cars
remaining cars	How many cars are in the approach. 200 m upstream from the stop line, including cars in the left-turn bay.	cars

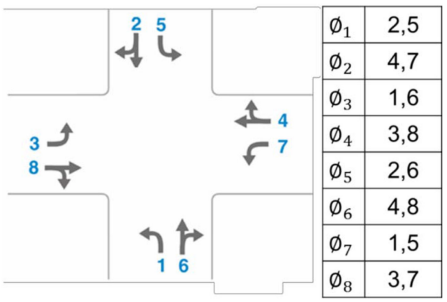

Fig. 4 *Phases configuration*

3.1 Environment, state, action and reward

The environment is a Markov process, and a reinforcement learning agent learns by interacting with its environment. In this case, our environment is an isolated signalised intersection as shown in Fig. 2.

Obviously, we cannot let the agent learn directly from the real world. Therefore, the environment is established in the microsimulator VISSIM which is a world-renowned traffic simulation software. Approach to the intersection is two lanes plus one left turn bay (about 20 m). The arrival rates of four arms are random in each episode but the sum of them will be a constant for maintaining a demand baseline. All the movement ratios are also constant. Random number seeds will also be changed for each episode to guarantee the randomness in every simulation.

Input data is the description of the current state. In general, using more information to describe a state can help network determine more precisely how much value the state is. One of the state descriptions is cell occupancy [16, 17]. It works like Fig. 3.

For each lane approaching the intersection is divided into many cells. If a cell is occupied by a vehicle, then the cell will return 1 to represent a car is in this cell, otherwise, return 0. This state description method provides a lot of information to the agent and almost resembles an image data input. Therefore, this method is expected to have a perfect combination with an image recognition vehicle detector. However, the larger the state size is, the more complicated the model is built. That is to say, it is quite possible that the model will be too hard to train and also consume a lot of computing resources. In this paper, we believed some specific traffic parameters would be enough to describe the difference between the states. These specific traffic parameters can be found in Table 1. For some parameters, the reason why they were selected was quite obvious. Remaining cars represent traffic demand; left-turn bay occupation pictures the possibility of spillback; current phase indicates the traffic signal state. However, for others parameters, it's not intuitive to select them as the elements in the state. In other words, some parameters, green and red durations, were selected based on experience.

The action here means to choose a signal phase. The possible signal phase configuration has been set as shown in Fig. 4. The agent will pick up one of these eight phases once a second. The inter-green limitation and minimum green time are taken into

Deep Q-learning Network
with memory replay and ε-greedy

Initialize replay memory D to capacity N
Initialize action-value function Q with random parameters θ
Initialize target action-value function $\hat{Q}$ with parameters $\hat{\theta} = \theta$
for episode = 1,M do
    Initialize environment and s_t
    for t=1,T do
        With probability ε select a random action a_t
        Otherwise select $a_t = \max_a Q^*(s_t, a; \theta)$
        Execute action a_t in environment and observe reward r_t and next state s_{t+1}
        Store transition (s_t, a_t, r_t, s_{t+1}) in D
        If the size of D is larger than N then
            Popout the oldest data in D
        Sample random minibatch of transition (s_j, a_j, r_j, s_{j+1}) from D
        $y_i = \begin{cases} r_j & for\ terminal\ s_{t+1} \\ r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a'; \hat{\theta}) & for\ non-terminal\ s_{t+1} \end{cases}$
        Perform a gradient descent step on loss function with respect to the network parameters θ
        Every C steps reset $\hat{Q} = Q$
    end for
end for

Fig. 5 *Algorithm 1: DQN with memory replay and ε-greedy*

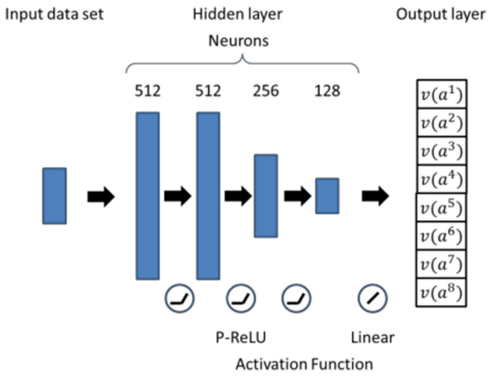

Fig. 6 *DQN model architecture*

consideration. It means if the agent decides to change the current phase, then in the following period of time, the agent will not able to make any new action. On the other hand, if the agent decides to maintain the current phase, it can just select the same signal phase. This implies the agent needs to learn the consequences of different actions in different states.

The reward is a guide which allows the agent to know whether it is moving in the right direction. In this case, we chose system delay as our rewards. Since the agent is doing a sequential decision-making process, for each state-action will generate an instant reward called r_t. Here we use time-step accumulated system delay as r_t. It means r_t is the accumulated system delay between the two actions. It's obviously by the terminal time T, the total reward R will equal to accumulated system delay

$$R = \sum_{t=1}^{T} r_t \tag{1}$$

Since the agent was seeking a policy to gain maximum total rewards, we turned all the r_t into a negative value. By doing so, we could expect a well-trained agent can make the intersection get a minimal total system delay.

3.2 DQN model

The basic concept of DQN is to find an equation Q in which inputs current state and will output each action's expected total reward. The optimal action-value function with the environment ε could be expressed using Bellman's equation as

$$Q^*(s, a) = \mathbb{E}_{s' \sim \varepsilon}[r + \gamma \max_{a'} Q^*(s', a') | s, a] \tag{2}$$

Here, a neural network is used as a non-linear approximator to estimate the action-value function. To train the neural network, TD error is considered as the loss. The loss function is expressed as

$$L_i(\theta_i) = \mathbb{E}_{s, a \sim \rho(\cdot)}\left[(y_i - Q(s, a; \theta_i))^2\right] \tag{3}$$

where θ stood for the parameters, $y_i = \mathbb{E}_{s' \sim \varepsilon}[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) | s, a]$ was the target for iteration i, and $\rho(s, a)$ is a probability distribution over sequences. Finally, we did a gradient descent with respecting θ and updating parameters. The algorithm of DQN is shown in Algorithm 1 (see Fig. 5).

There are two key tips in the algorithm: memory replay [19] and target network [8]. Memory replay allows an agent to learn from earlier experience and speeds up learning. Target network helps agent learn the Q function more stably. The model architecture is shown in Fig. 6.

3.3 Value-based state-action model

A classic DQN model will approximate every action's value in the current state. However, during the iteration of using TD error as loss function, it only receives one error from one $\{s_j, a_j, r_j, s_{j+1}\}$ data. In this case, only one action-value would be updated and the remaining seven action-value would not have any change since they were estimated by the model itself in the first place.

We believed this feature would cause an inefficient result in the training process. Instead of outputting multi action-value, we thought actions as part of the input layer and turn the output into a single value. The model architecture is shown in Figs. 7.

Some sequences need to be modified in the original algorithm. First, we needed to list all the possible actions and feed them into the model to receive values. Then we chose the action that leads maximum value as our policy.

Since the actions here were limited discrete set, it was feasible to list all the action-value and compared them to each other.

The second modification is about discount factor γ. As mentioned in Section 3.1, our action is to select phases. There is an execution time difference between actions and can be revealed in the following illustration (Fig. 8).

The illustration is a simple action based case in which has only two actions: upper and lower. Taking upper action needs 1 time-step and lower needs 2. If we use the function in Algorithm 1 (Fig. 5), $y_i = r_i + \gamma \max_{a'} \hat{Q}\left(s_{j+1}, a'; \hat{\theta}\right)$, to do iteration, we can find

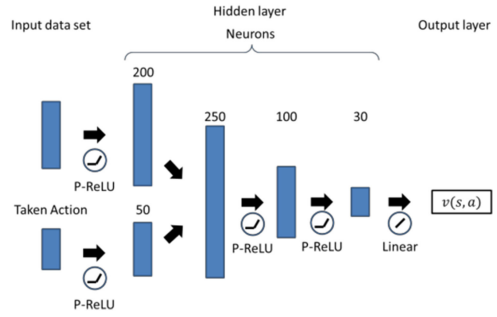

Fig. 7 *State-action model architecture*

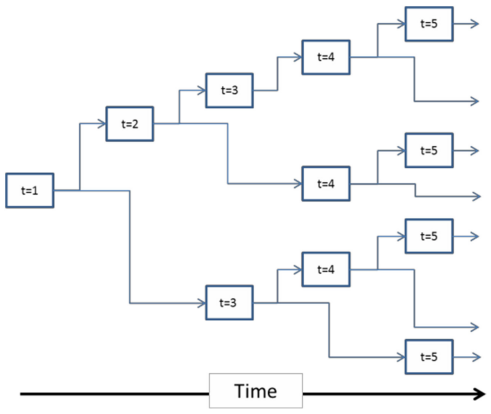

Fig. 8 *Time difference between actions' execution*

Table 2 Training settings

Parameter	Value
simulation times	600 s
network parameter update rate	1/episode
memory replay capacity: N	30,000
episode	200
ϵ	0.1
ϵ decrease rate	0.001/episode
discount factor γ	0.99
arrival rate	$\sim N(800,100)$ pcu
optimiser	Adam [20]
initialisation	Glorot Initialisation [21]
loss function	mean square error of TD
Abmer period	3 s
all red	3 s

a fact that the bottom state while $t=5$ has only been discounted two times. On the contrary, the top state while $t=5$ has been discounted five times already. This simple case demonstrates that regardless of the relation between reward and action, the action-based structure will still lead Q value to a bias estimation.

Therefore, we suggested that the action execution time E needed to be considered into the discount factor. The execution time E is defined as the time interval between two decisions in a row. δ is the dynamic discount factor depend on E and original fixed discount factor γ

$$\delta = 1 - E(1-\gamma) \tag{4}$$

$$y_i = r_i + \delta \max_{a'} \hat{Q}\left(s_{j+1}, a''; \hat{\theta}\right) \tag{5}$$

The dynamic discount factor in (4) was derived from the sum formula of infinite geometric series. By using (5), we can ensure that at each time-step the Q value will have the same level of discount. We supposed this modifying would encourage agent more inclined to maintain the same phase than fixed γ since the prior methods would repeat reducing Q value for each action.

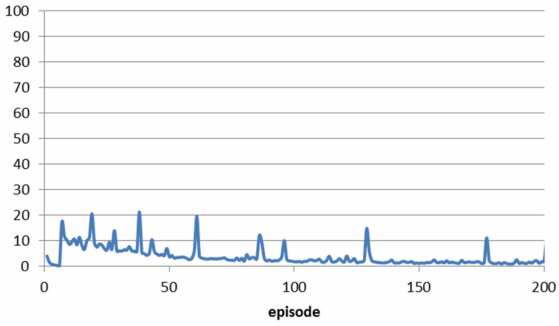

Fig. 9 *DQN model average loss in episode*

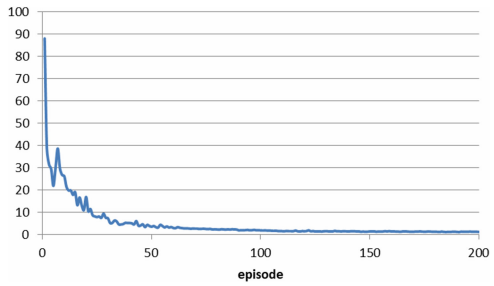

Fig. 10 *S-A model average loss in episode*

Maintaining the same phase means more actions during a certain period of time and means more discount than changing a phase.

4 Training and scenario testing

4.1 Training

The performance of both traditional DQN and S-A model was evaluated with traffic metrics: system total delay and average delay per car. In order to know whether the agent has converged the Q value function, the loss value over episodes is also observed. Each episode is set to simulate 600 sim seconds from the initial state. After each episode, the target network will update the same parameter as a model. The rest of the settings are shown in Table 2.

Particularly worth mentioning, before starting training, data was collected under a random action policy. The purpose of doing so is to make sure agent was learning from a stochastic experience instead of the initial network parameters. The data was collected and stored in the memory replay space. The agents will have the opportunity to re-use these experiences before they are removed.

Two models' training was processing in the python script and used Tensorflow as backend. The data during training is shown in Figs. 9–12, respectively, DQN model's loss over episodes, S-A model's loss over episodes, DQN model's performance over episodes and S-A model's performance over episodes.

From the run chart of loss and performance, it indicates that both models reached convergence around episode 100. The performances trend reached a stable value after 150 episodes. In contrast to performance, the S-A model seems to be more stable than traditional DQN model and also gains a better performance at the end of training.

4.2 Scenario testing

In this part, we set up two arrival rate scenarios to test the models' performance: unsaturated and oversaturated. The arrival rates were given 800 pcu/hr for each approach in an unsaturated scenario which was the same total demand as training setting. In the oversaturated scenario, we gave 2000 pcu/hr input on North and West boundary, and 100 for East and South. Turning movement ratios remained the same as what they were during training (Fig. 13).

We compared the performance of the three control methods: Fix Timing Plan, traditional DQN model and S-A model. The fix timing plans were calculated by the delay formulas from Webster

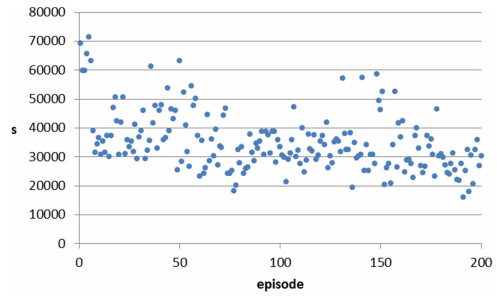

Fig. 11 *DQN model system total delay in episode*

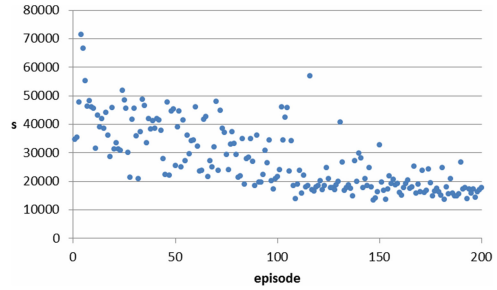

Fig. 12 *S-A model system total delay in episode*

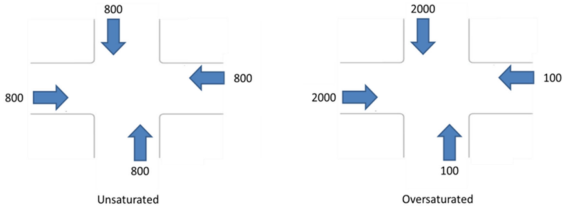

Fig. 13 *Scenarios and demand setting*

[22]. While dealing with the oversaturated intersection, traffic engineer usually used maximum cycle length to optimise splits. Here we used 180 s as our maximum cycle length.

System total delay is the first evaluation index since it is the reward during the training. And also it is the most common traffic performance indicators. The second evaluation index is total throughput. In the oversaturated scenario, it's more common to take throughput as performance instead of delay because it will be hard to detect a keep growing queuing in practice (although it is totally fine in the simulation).

For each case, simulations ran five times with different random number seed to make sure the variability in scenarios. Comparing results are shown in Fig. 14.

4.3 Discussion

Fig. 15 shows that the S-A model can improve performance by 20 and 15%, respectively, to unsaturated and oversaturated scenarios. The average throughput in the oversaturated scenario is 17% higher than the optimised fix timing plan. On the other hand, traditional DQN model could not get higher performance than optimised fix timing plan in the unsaturated scenario, and almost the same level on average throughput in the oversaturated scenario.

The S-A model has achieved a significant effect on traffic signal control. We found in this study that the agent would learn some phase sequences to gain better results, such as emptying left turn bay before through phase (∅5 and ∅6) if there might be overflowed. And the agent could extend phase timing if the demand was still existing. An interesting find in DQN policy is that it seems to change phase at each state. In other words, it thinks extending the current phase is not the best choose on the Q value. This phenomenon proved the idea about the discount factor γ in Section 3.3.

Furthermore, compared to the training loss between Figs. 9 and 10, loss decreases more stable in the S-A model than the DQN model. The reason for this behaviour is still not clarified with

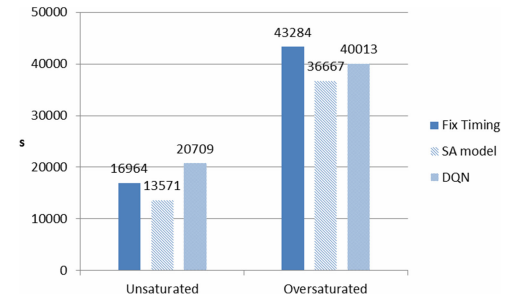

Fig. 14 *Average system total delay*

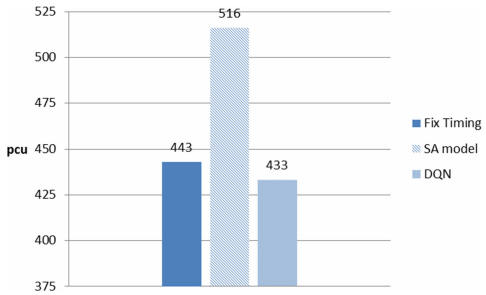

Fig. 15 *Oversaturated average throughput*

theoretical analysis in this paper, but we suppose the explanation is because of the discount factor γ. Dynamic discount factor helps agent makes an unbiased estimate, thus it leads the accuracy of estimation to improve in a more gradually way.

In this research, though we only used five traffic parameters (56 elements in total) to describe the state of the intersection, it did not imply these parameters were the only requirement for the DRL model. On the contrary, we believed more information was always a good thing to a deep neural network once it's trainable. The parameters we used here could be combined with cell occupancy to describe the state more detail but also maintain some domain knowledge.

The fact that DRL signal control works in both scenarios is really an exciting discovery. It means that this method is a real online adaptive control, and it's very likely to be a universal solution for signal control. However, yet, there still remain lots of factors which are not taken into consideration. Just mentioning in the last paragraph, the phase sequence is learned by the agent itself, and it's very different from people's common sense. For example, drivers usually default a left turn phase will follow through phase behind (or before). However, the agent only concerns what phase will gain the best performance and thence the outcome is completely different from the driver's experience.

Another important issue is fairness. The fairness is not taken into consideration in this model, but it really matters in practical signal timing design. It also relates to drivers' mood: they never know how long they have to wait. More and more cities installed red light countdown timer at the intersection. Studies show that red light countdown timer can help reduce both start delay and the probability of pedestrian running a red light. In this DRL control method, which phase will be the next and for how long are only known by then. This characteristic might lead to the result that more drivers try to run a red light.

The above section describes some of the difficulties that may be encountered in practical applications. But still, the potential of DRL signal control is no doubt.

5 Summary and further work

A value-based state-action function was proposed in this paper. The method is mainly based on DQN's algorithm with a novel design both in neural network architecture and time-dependent discount factor to reflect the cycle-free strategy in traffic signal control.

Training and scenario testing had conducted. The results show that the S-A model works in both scenarios: unsaturated and

oversaturated. The S-A model can improve the performance by 15–20% compared to an optimised fix timing plan. And it's more stable than the traditional DQN model on the training process and the different scenarios testing. This result might imply the discount factor γ needs to be modified if the actions have different execution time. In addition, the experience from this research suggested that regarding actions as one part of inputs and outputting only one single value might be easier to the train a DRL model.

Future work will keep focusing on applying DRL to a multi-intersection. DQN framework can only be used while the action set is discrete. The combination of multi-intersection signal phase is kind of a continuous actions problem. There are several DRL algorithms designed to solve a continuous action task, such as actor-critic, DDPG and PPO. How to apply these state-of-the-arts to traffic signal control will be one of the hottest topics in the next generation ITS.

6 Acknowledgments

This paper was supported by China Engineering Consultants, Inc. (http://www.ceci.org.tw/) and condensed from project no. 06923. The authors thank Dr. Hung-yi Lee, assistant professor in the Department of Electrical Engineering, National Taiwan University, for his valuable communication about training techniques of deep learning. All the programmers of open software used in this study are also highly appreciated.

7 References

[1] Sutton, R.S., Barto, A.G.: '*Reinforcement learning: an introduction*' (The MIT Press, Cambridge, Massachusetts, USA/London, England, 1998)

[2] Kaelbling, L.P., Littman, M.L., Moore, A.W.: 'Reinforcement learning: a survey', *J. Artif. Intell. Res.*, 1996, **4**, pp. 237–285

[3] Bertsekas, D.P.: '*Dynamic programming and optimal control*' (Athena Scientific, Nashua, NH, USA, 2007, 3rd edn.), vol. **2**

[4] Abdulhai, B., Kattan, L.: 'Reinforcement learning: Introduction to theory and potential for transport applications1', *Can. J. Civ. Eng.*, 2003, **30**, pp. 981–991

[5] Abdulhai, B., Kattan, L.: 'Reinforcement learning: introduction to theory and potential for transport applications', *Can. J. Civ. Eng.*, 2003, **30**, pp. 981–991

[6] Watkins, C.J.C.H.: 'Learning from delayed rewards', Ph.D. thesis, Cambridge University, 1989

[7] Watkins, C., Dayan, P.: Q-learning, *Machine Learning*, 1992, **8**, (3-4), pp. 279–292

[8] Mnih, V.*, et al.*: 'Human-level control through deep reinforcement learning', *Nature*, 2015, **518**, (7540), pp. 529–533

[9] Bengio, Y.: 'Learning deep architectures for AI', *Found. Trends Mach. Learn.*, 2009, **2**, pp. 1–127

[10] Krizhevsky, A., Sutskever, I., Hinton, G.: 'Imagenet classification with deep convolutional neural networks', *Adv. Neural Inf. Process. Syst.*, 2012, **25**, pp. 1106–1114

[11] Hinton, G.E., Salakhutdinov, R.R.: 'Reducing the dimensionality of data with neural networks', *Science*, 2006, **313**, pp. 504–507

[12] Abdulhai, B., Pringle, R., Karakoulas, G.J.: 'Reinforcement learning for true adaptive traffic signal control', *J. Transp. Eng.*, 2003, **129**, (3), pp. 278–285

[13] Wiering, M.: 'Multi-agent reinforcement learning for traffic light control'. ICML, Stanford, CA, USA, 2000, pp. 1151–1158

[14] Bingham, E.: 'Reinforcement learning in neurofuzzy traffic signal control', *Eur. J. Oper. Res.*, 2001, **131**, (2), pp. 232–241

[15] Mnih, V., Kavukcuoglu, K., Silver, D.*, et al.*: 'Playing atari with deep reinforcement learning', arXiv preprint arXiv:1312.5602, 2013

[16] Genders, W., Razavi, S.: 'Using a deep reinforcement learning agent for traffic signal control', Available at: https://arxiv.org/abs/1611.01142, November 2016

[17] van der Pol, E.: 'Deep reinforcement learning for coordination in traffic light control', Master's thesis, University of Amsterdam, August 2016

[18] Li, L., Lv, Y., Wang, F.-Y.: 'Traffic signal timing via deep reinforcement learning', *IEEE/CAA J. Autom. Sinica*, 2016, **3**, (3), pp. 247–254

[19] Lin, L.-J.: 'Self-improving reactive agents based on reinforcement learning, planning and teaching', *Mach. Learn.*, 1992, **8**, (3–4), pp. 293–321

[20] Kingma, D.P., Ba, J.: 'Adam: a method for stochastic optimization', arXiv:1412.6980 [cs.LG], December 2014

[21] Glorot, X., Bengio, Y.: 'Understanding the difficulty of training deep feedforward neural networks'. Proc. of the Int. Conf. on Artificial Intelligence and Statistics. Society for Artificial Intelligence and Statistics, 2010, vol. 9, pp. 249–256

[22] Webster, F.V.: 'Traffic Signal Settings'. Road Research Technical Paper No. 39. London: Great Britain Road Research Laboratory, 1958