International Journal of Advances in Computer Science & Its Applications – IJCSIA

Volume 9 : Issue 2- [ISSN : 2250-3765] - Publication Date: 27 Dec, 2019

DEEP REINFORCEMENT LEARNING BASED TRAFFIC CONTROL SYSTEM

Bryan Chua Seck How, Kamarulafizam Ismail*, Ng Ting Sheng, Fazila Mohd Zawawi

***Abstract*— Existing traffic light controls are ineffective and causes a handful of problems such as congestion and pollution. The purpose of this study is to investigate the application of deep reinforcement learning on traffic control systems to minimize congestion at a targeted traffic intersection. The traffic data was extracted, analyzed and simulated based on the Poisson Distribution, using a simulator, Simulation of Urban Mobility (SUMO). In this research, we proposed a deep reinforcement learning model, which combines the capabilities of convolutional neural networks and reinforcement learning to control the traffic lights to increase the effectiveness of the traffic control system. The paper explains the method we used to quantify the traffic scenario into different matrices which fed to the model as states which reduces the load of computing as compared to images. After 2000 iterations of training, our deep reinforcement learning model was able to reduce the cumulative waiting time of all the vehicles at the Pulai Perdana intersection by 47.31% as compared to a fixed time algorithm and can perform even when the traffic is skewed in a different direction. When the traffic is scaled down to 50% and 20 %, the agent continues to improve the waiting time by 69.5% and 68.36 % respectively.**

Keywords*— *Traffic Light Control, Deep Reinforcement Learning, SUMO

I. Introduction

The creation of traffic lights creates an equal opportunity to cross an intersection, but conventional traffic control systems only causes traffic congestion, which impedes the flow and causes many problems for the general commuters. Traffic jams are often associated with lost in productivity, frustration and accidents. It has also led to several serious social problems such as long travelling times, increased fuel consumption and air pollution. (Gao, Shen, Liu, Ito, & Shiratori, 2017). According to another study done by Boston Consulting Group (BCG) known as "Unlocking Cities", they showed that drivers in Kuala Lumpur spend about 53 minutes stuck in traffic jams every day. That roughly sums up to 13.4 days in total spent in traffic in a year.

Bryan Chua Seck How
School of Mechanical Engineering, Universiti Teknologi Malaysia Malaysia

Kamarulafizam Ismail[1,2]
[1]Media and Game Centre of Excellent, Institute of Human Centered Engineering, Universiti Teknologi Malaysia
[2]School of Mechanical Engineering, Universiti Teknologi Malaysia Malaysia

Traffic problem is a very complex issue since it involves many parameters. Firstly, it is heavily dependent on the time of day and week, general during rush hours, which is in the morning or afternoon, the traffic flow is severely increased because users need to get to or off work. Weekends generally show a decrease in traffic loads. Secondly, existing traffic light control either deploys fixed programs without considering real-time traffic or considering the traffic to a very limited degree (Liang, Du, Wang, & Han, 2018).

Adaptive traffic signal control, which adjusts traffic signal timing according to real-time traffic, has been shown to be an effective method to reduce traffic congestion. With recent advancements in Machine Learning technology, many researchers have shown interest in the capabilities of Deep Learning and Reinforcement learning since they are able to be able to learn through a large set of data unsupervised. In recent developments, we can see machine learning algorithms being able to surpass human level intelligence in the game of AlphaGo.

Recently, more and more studies on smart traffic light control system have been conducted. Many researchers now believe that machine learning algorithms can improve traffic light control and management. Furthermore, with the advancements in hardware and algorithms within the recent years, deep learning algorithms are also being experimented as well, this will be discussed in depth within the report. Generally, fixed time traffic signals are being deployed in urban area`s due to its regularity and predictability. Some traffic signals deliberately stop drivers from experiencing a string of green lights, thus discouraging high volumes of traffic while still preventing congestion. Inductive loops, as seen in the figure above, are generally used to keep traffic flowing in the main roads of traffic and to detect if there are vehicles waiting to cross from the side roads. It can also reduce waiting time at the traffic intersection. It can also be used to change or lengthen traffic light phases if the que is longer.

In terms of Deep Reinforcement Learning, Li, Lv, and Wang (2016) proposed to use a deep stacked autoencoders (SAE) neural network to estimate the Q function, the neural net can take massive amounts of input states and return the possible Q value for each possible action. Genders and Razavi (2016) has shown that convolutional neural networks (CNN) can be used to approximate the optimal Q values. One of the most obvious contribution from their study is the use of discrete traffic state encoding (DTSE) as a better representation of traffic information. (Liang et al., 2018)

further improved the use of deep reinforcement learning in traffic light controls by introducing Double Dueling Deep Q Networks called (3DQN).

This research propose a deep reinforcement learning algorithm that can extract all key features useful for adaptive traffic signal control from raw real-time traffic data. By extracting useful features such as the position and speed of the cars and allow further process of the data by the deep reinforcement learning algorithm, we will be able to make proper decisions that will allow traffic to be monitored and controlled more effectively. With deep reinforcement learning, the traffic lights will be able to learn from real time traffic conditions and make decisions better from previous sample data it has collected, which makes it more "experienced" it's management of the traffic junction the longer it operates.

II. Methodology

The simulation tool, Simulation of Urban Mobility (SUMO), will be used to simulate the junction at Pulai Perdana, a heavily congested intersection in Johor Bahru, Johor as accurately as possible. Python will also be utilized to interface with the simulation software and deploy deep reinforcement learning to actuate the traffic signals. In Python, the deep learning library Keras will also be used to allow the algorithm to learn from its actions.

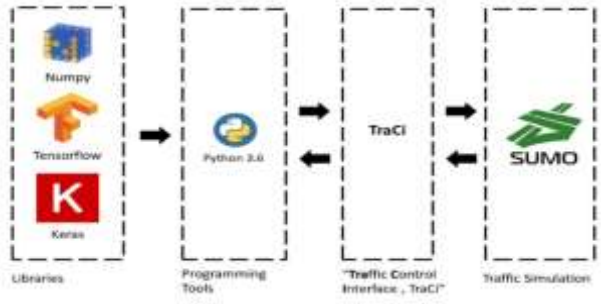

Figure 1. Simulation Software Architecture.

A. *Problem Definition*

In the table shown below, LE1 depicts traffic coming from Pontian where are LE3 shows traffic from Skudai. LE2 on the other hand shows traffic coming from Persiaran Pulai Perdana and LE4 shows traffic coming from Jalan Teratai. Each junction will be simulated to have 3 lanes in the departure junction and 2 lanes in the arrival junction as shown in Figure 2. The traffic data was obtained for an hour at the said intersection.

TABLE I. TRAFFIC INFORMATION AT PULAI PERDANA INTERSECTION

Junction	Number of cars per hour	Green Signal Duration(s)
LE1	1076	100
LE2	796	38
LE3	1345	79
LE4	581	38

B. *Vehicle Arrival Process*

The traffic conditions are simulated based on Mathew (2014), which shows the method of simulating traffic flow through the use of random variates that follows the Poisson distribution to generate vehicles that arrives in a given time interval so that it follows a typical vehicle arrival process. In the SUMO simulation software, the traffic information is read from the route.xml file.

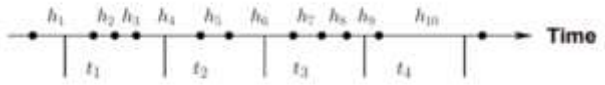

Figure 2. Illustration of vehicles arriving modelling.

$$p(x) = \frac{\mu^x e^{\mu}}{x!} \quad (1)$$

Equation (1) is the probability of the density function.

C. *States*

The studies done by Genders and Razavi (2016), they utilized their discrete traffic state encoding (DTSE) method which allows them to retain useful traffic information. The agent will observe the states to be St= (P, V, L) ∈S for signal control. The states will then be used as what the DQNAgent "sees", the environment encoded into a matrix for the agent to make sense of the environment and make decisions based on the states. Figure 3 shows how the agent observes the environment, the agent creates a Boolean value of 1 when it detects a car if present within the cell length, the velocity matrix is also obtained by dividing the actual speed of the vehicle with the max allowable speed. The states allow the agent to perceive the entirety of the environment through the use of matrices instead of whole images in efforts to reduce computational difficulty.

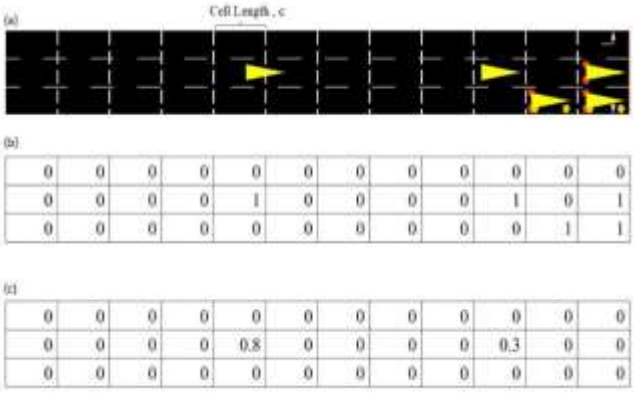

Figure 3. (a) Example of simulated traffic (b) with corresponding Boolean (c) and real-valued velocity vectors.

$$\text{positionMatrix} = \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}, \; velocityMatrix = \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix}, \; \text{lghts} = \begin{bmatrix} L_0 \\ L_1 \\ L_2 \\ L_3 \end{bmatrix}$$

D. *Convolutional Neural Network*

After observing the states, the agent will be able to take an action based on what it "sees". The process of seeing involves the use of a Convolutional Neural Network that allows extraction of important features from the state matrices.

The input states or the agent's observed states are positionMatrix, VelocityMatrix and lgts. The first layer of convolution has 16 filters of 4x4 with stride of 2 and it uses ReLU (Rectified Linear Units) as the activation function. The second layer has 32 filters of size 2x2 with a stride of 1 and uses ReLU. The 3rd and 4th layers are fully connected layers with a size of 128 and 64 respectively. The final layer is then a layer with a linear output that outputs the Q value that corresponds to every possible action.

E. *Action*

When the green light interval ends, the current time step t ends and a new time step begins. The agent then proceeds to observe a new time step and chooses the next action. The same actions may be chosen across time steps, which will cause the green light interval to run again for another 10 seconds. However, if the action selected is different from the previous action, which is to change the traffic signals. The yellow lights will be actuated for the junction for 3 seconds before actuating the green light. Since the agent's goal is to reduce the overall waiting time, the agent will need to find an action policy as denoted in (E) that maximizes the following cumulative future rewards. . After observing a given state, the agent decides to take an action based on an action policy π.

F. *Rewards*

One of the biggest differentiators between reinforcement learning and other learning algorithms is the rewards. Rewards functions as a feedback system to allow the model to access its performance based on its previous actions. Since the main goal is to see if the model can increase the efficiency of the traffic light control system, the main parameters that can best reflect the efficiency is the vehicle waiting time. Thus, we define rewards as the difference in cumulative waiting time between active and number of vehicles previously in the inactive traffic, where r1 is the cumulative number of vehicles at a given active junction and r2 is the cumulative waiting time of idle vehicles waiting at the inactive junction.

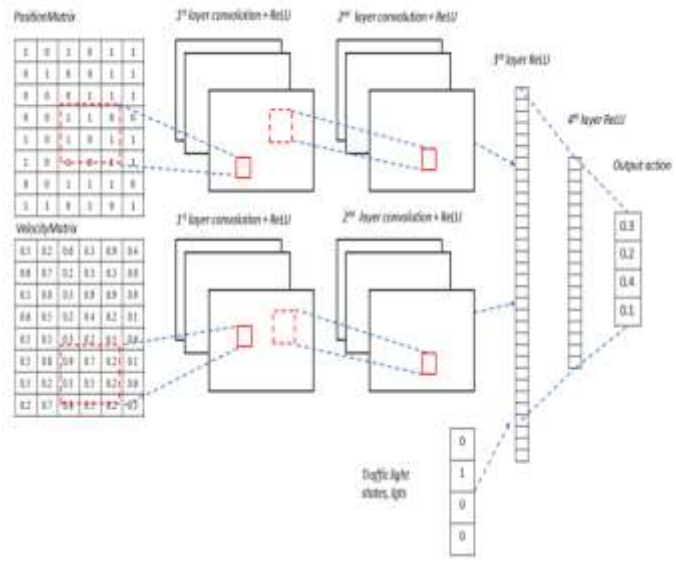

Figure 4. Convolutional neural network approximating the Q values.

TABLE II. TRAFFIC LIGHT PHASES FOR SIMULATED LANES.

	LE1			***LE2***			***LE3***				***LE4***	
	L0	***L1***	***L2***	***L0***	***L1***	***L2***	***L0***	***L1***	***L2***	***L0***	***L1***	***L2***
Phase 0	***G***	***G***	***G***	*r*	*r*	*r*	*r*	*r*	*r*	*r*	*r*	*r*
Phase 1	***y***	***y***	***y***	*r*	*r*	*r*	*r*	*r*	*r*	*r*	*r*	*r*
Phase 2	*r*	*r*	*r*	***G***	***G***	***G***	*r*	*r*	*r*	*r*	*r*	*r*
Phase 3	*r*	*r*	*r*	***y***	***y***	***y***	*r*	*r*	*r*	*r*	*r*	*r*
Phase 4	*r*	*r*	*r*	*r*	*r*	*r*	***G***	***G***	***G***	*r*	*r*	*r*
Phase 5	*r*	*r*	*r*	*r*	*r*	*r*	***y***	***y***	***y***	*r*	*r*	*r*
Phase 6	*r*	*r*	*r*	*r*	*r*	*r*	*r*	*r*	*r*	***G***	***G***	***G***
Phase 7	*r*	*r*	*r*	*r*	*r*	*r*	*r*	*r*	*r*	***y***	***y***	***y***

$$r_t = r_1 - r_2 \tag{2}$$

The reward is then calculated after the agent finishes its action step, which in this case, the reward will be calculated after the 10 second period of actuation of the green light. Then the reward will be reset to zero once the traffic agent changes the phase and restarted once again.

G. *Agent Hyperparameters*

The greedy epsilon algorithm is deployed, where the value of ε is 1.0 in the beginning to assume explorative behaviors, however, the value of epsilon starts to decay at a rate of 99.5% every single time the states are observed until it reaches the minimum value of 0.01, where the agent starts to change from taking explorative actions to exploitative one. The discount factor for future rewards is set at 0.95. The optimizer selected will be the Root Mean Squared Prop (RMSProp) algorithm, which uses a moving average of squared gradients to normalize the gradient itself, the algorithm is a stochastic technique for mini-batch learning. The learning rate for the RMSProp algorithm is set at 0.0002 for optimal results as mentioned in the previous chapter. The capacity of replay memory is also set at 200 to minimize memory usage.

H. *Agent Training*

The agent will be trained for 2000 episodes, each episode corresponds to 1 hour. We first initialize the neural network with random weights. At the start of each time step, the agent observes the current time step St and the input is fed into the neural network and performs an action at that will provide the highest cumulative future reward. The agent then receives a reward Rt and proceeds to obtain the next step St+1 in the environment. These information (St, At, Rt, St+1) are stored as experiences in its memory. As the memory is limited in size, the oldest data is deleted when the memory is full. The DNN is then trained by extracting training examples from the memory. This is known as experience replay. The agent then proceeds to learn features \theta, by training the DNN network to minimize the following Mean Squared Error (MSE), as in

$$MSE(\theta) = \frac{1}{m}\sum_{t=1}^{m}\left\{\left(R_t + \gamma \max_{a'} Q(S_{t+1}, a'; \theta')\right) - Q(S_t A_t; \theta)\right\}^2 . (3)$$

Since m is the size of the input data set, which in our case is very large, it would be very computationally expensive to calculate. Hence, we will use the stochastic gradient descent algorithm RMSProp with a minibatch of 32 as mentioned before.

III. Results & Discussion

By examining our simulation data shown in figure 5 & 6, we were able to show that the algorithm is indeed in the right path in learning a good action selection policy that effectively reduces the cumulative vehicle waiting time at the traffic lights. Our algorithm's results start to converge midway through the episodes and becomes more stable.

During the training, the minimum cumulative waiting time achieved was 115882 seconds. The average value of the cumulative waiting time of all vehicles at the junction is 203221 second.

At 200 episodes, we see the waiting time of vehicles at a junction gradually reducing as the agent finds suitable action policies that allows it to make better decisions. The spikes in the graphs shows that the explorative nature of the agent allows it to try out different actions, not necessarily resulting in reduction in waiting time but crucial for exploring different actions that may give positive results. At 800 episodes, we see the results start to converge and the waiting time starts to stabilize from this episode onwards. The stabilizing mechanisms such as the experience replay is proven to be effective in stabilizing the action selection policy.

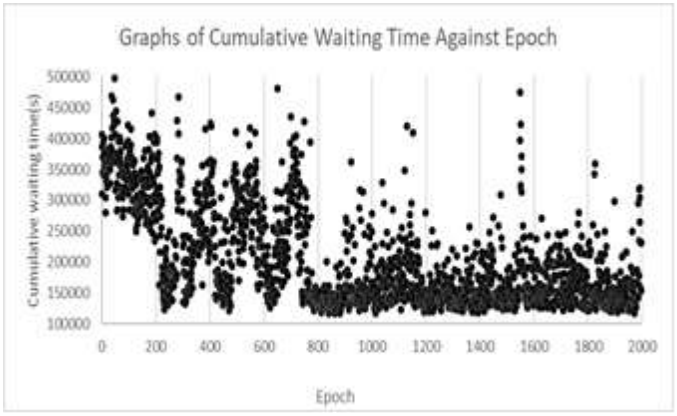

Figure 5. Graph of Cumulative Waiting Time Against Epoch

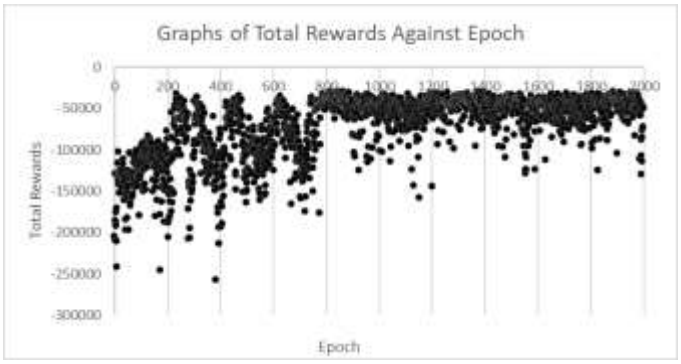

Figure 6. Graph of Total Rewards Time Against Epoch

After running the training for 2000 episode, the agent learnt a good action selection policy and managed to reduce the cumulative waiting time. The agent is now used to run the simulation once again using several carrying traffic conditions and compared to the fixed time algorithm. The agent is tested on high traffic conditions, high traffic conditions with traffic skewed to another direction, medium traffic conditions and low traffic conditions in comparison with the fixed time algorithm to evaluate its performance improvement as compared to the fixed time algorithm.

The skewed traffic will be simulated by adjusting the heavy traffic to lanes LE2 and LE4 instead of LE1 and LE3. The medium traffic and low traffic are assumed at 50% and 20% of the high traffic volume. The next table shows the simulated result using the final weights of the algorithm after 2000 episodes of training.

TABLE III. CUMULATIVE WAITING TIME FOR DIFFERENT ALGORITHMS AND TRAFFIC HEAVINESS.

Traffic Heaviness	Number of cars in one hour in given lane				Cumulative Waiting Time (s)	
	LE1	LE2	LE3	LE4	Agent	Fixed time algorithm
High	1076	796	1345	581	213696	405587
High (skewed traffic)	581	1076	796	1345	214522	704820
Medium	538	398	673	291	48592	153566
Low	215	159	269	116	18206	59549

Based on the results shown above, it is clear that the agent outperforms the fixed time algorithm in every type of traffic heaviness, where in high traffic conditions, high traffic conditions with skewed traffic, medium traffic conditions and low traffic conditions, there is a 47.31 %, 69.56% ,68.36 % and 69.43 % reduction in waiting time respectively. The fixed time algorithm fairs well at high traffic conditions, however, as the signal timing is set to meet the demand of the intersection, it's performance greatly reduces when the traffic is skewed to another direction. Although the number of cars passing through the intersection is the same, the fixed time algorithm can't handle the change in direction of the traffic heaviness as it is preset to a certain timing algorithm. The agent on the other hand is adaptive to traffic and was able to keep the cumulative waiting time a constant value of 213696s and 214522s. This show's that the agent is adaptive and can execute a traffic control policy to solve the current traffic conditions.

When the traffic heaviness is reduced to 50% and 20% respectively, we can see that the performance of the agent is 68.36% and 69.43% better than the fixed time algorithm. The performance is also better as compared to the high traffic conditions. The fixed time algorithm is not suited to adapt to the everchanging traffic heaviness, especially at lower traffic conditions.

IV. Conclusion

In this paper, we proposed to solve the traffic light control problem at the Pulai Perdana intersection using a deep reinforcement learning model. This research study was a success with all objectives achieved. The research started with the notion that artificial intelligence could one day be used as an agent to manage a traffic intersection by controlling the traffic lights. It is proven in the experiment that a deep reinforcement learning model was able to reduce the cumulative waiting time of all the vehicles at a given traffic intersection as compared to a fixed time algorithm-based traffic management system at Pulai Perdana by 47.31%.

V. Acknowledgment

The author would like to thank Universiti Teknologi Malaysia for funding this research through Research University Grant (RJ130000.2424.03G98) and the ministry of Education Malaysia

VI. References

[1] Gao, J., Shen, Y., Liu, J., Ito, M., & Shiratori, N. (2017). Adaptive Traffic Signal Control: Deep Reinforcement Learning Algorithm with Experience Replay and Target Network. arXiv preprint arXiv:1705.02755.

[2] Genders, W., & Razavi, S. (2016). Using a deep reinforcement learning agent for traffic signal control. arXiv preprint arXiv:1611.01142.

[3] Ghazal, B., ElKhatib, K., Chahine, K., & Kherfan, M. (2016). Smart traffic light control system. Paper presented at the Electrical, Electronics, Computer Engineering and their Applications (EECEA), 2016 Third International Conference on.

[4] Goel, A., & Kumar, P. J. A. E. (2015). Characterisation of nanoparticle emissions and exposure at traffic intersections through fast–response mobile and sequential measurements. 107, 374-390.

[5] Li, L., Lv, Y., & Wang, F.-Y. (2016). Traffic signal timing via deep reinforcement learning. IEEE/CAA Journal of Automatica Sinica, 3(3), 247-254.

[6] Liang, X., Du, X., Wang, G., & Han, Z. (2018). Deep reinforcement learning for traffic light control in vehicular networks. arXiv preprint arXiv:1803.11115.

[7] Mathew, D. T. V. (2014). Transportation Systems Engineering, Chapter 13: Vehicle Arrival Models: Count.

[8] Özlü, A. (2017). Vehicle Detection, Tracking and Counting. https://github.com/ahmetozlu/vehicle_counting

[9] Tahifa, M., Boumhidi, J., & Yahyaouy, A. (2015). Swarm reinforcement learning for traffic signal control based on cooperative multi-agent framework. Paper presented at the Intelligent Systems and Computer Vision (ISCV), 2015.

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

[11] Zhang, S., & Sutton, R. S. J. a. p. a. (2017). A deeper look at experience replay.