Smarter and Safer Traffic Signal Controlling via Deep Reinforcement Learning

Bingquan Yu
Tongji University, Shanghai, China
1931540@tongji.edu.cn

Jinqiu Guo
Tongji University, Shanghai, China
jqguo@tongji.edu.cn

Qinpei Zhao
Tongji University, Shanghai, China
qinpeizhao@tongji.edu.cn

Jiangfeng Li
Tongji University, Shanghai, China
jfli@tongji.edu.cn

Weixiong Rao
Tongji University, Shanghai, China
wxrao@tongji.edu.cn

ABSTRACT

Recently deep reinforcement learning (DRL) has been used for intelligent traffic light control. Unfortunately, we find that state-of-the-art on DRL-based intelligent traffic light essentially adopts discrete decision making and would suffer from the issue of unsafe driving. Moreover, existing feature representation of environment may not capture dynamics of traffic flow and thus cannot precisely predict future traffic flows. To overcome these issues, in this paper, we propose a DDPG-based DRL framework to learn a continuous time duration of traffic signal phases by introducing 1) a transit phase before the change of current phase for better safety, and 2) vehicle moving speed into feature representation for more precise estimation of traffic flow in next phase. Our preliminary evaluation on a well-known simulator SUMO indicates that our work significantly outperforms a recent work by much smaller number of emergency stops, queue length and waiting time.

CCS CONCEPTS

• **Computing methodologies** → *Control methods*; • **Applied computing** → *Transportation.*

KEYWORDS

Traffic Light Control; Deep Reinforcement Learning,

ACM Reference Format:
Bingquan Yu, Jinqiu Guo, Qinpei Zhao, Jiangfeng Li, and Weixiong Rao. 2020. Smarter and Safer Traffic Signal Controlling via Deep Reinforcement Learning. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3340531.3417450

1 INTRODUCTION

Recent years witness the rapid growth of various vehicles and traffic jam in urban cities [3]. Intelligent traffic light control offers the opportunity to optimize traffic flow, save travel time, and reduce traffic jam. Existing methods usually deploy traffic light control programs by setting fixed time duration in every cycle or dynamical time duration via historical traffic information or rather coarse traffic information detected by underground inductive loop detectors. Unlike these methods, the recently popular deep reinforcement learning (DRL)-based traffic light control programs are promising to overcome the deficiency of these existing methods by taking into account real-time traffic and setting adaptive time duration.

In the DRL-based traffic light control framework such as Intellilight [9], *environment* consists of traffic light phase and traffic condition. By treating traffic condition as an image, a CNN-based neural network can process the input image to learn features and then to enrich such learned features by hand-crafted features of the environment (e.g., the number of waiting vehicles, and queue length). *State* is then a feature representation of the environment. *Agent* next takes the state as input and learns a decision-making model to pick an *action* from a binary action space: either "keep" or "change" the current phase of traffic lights. The decision is sent to the environment and the *reward* (e.g., how many vehicles pass the intersection) is sent back to the agent. The agent consequently updates the model and further makes the new decision for the next phase based on the new state and updated model.

When processing real-time traffic condition, the DRL-based framework can intelligently control the time period of the entire sequential phases whose next phase changes (we call it a *stage*). Since each phase is with a fixed time interval (e.g., 5 seconds), the entire time duration of such a stage is times of the interval. For example, given 10 phases within a stage, the time duration of this stage is 50 seconds (= 10×5).

Unfortunately, the DRL-based framework above suffers from two following issues. First, it essentially makes the discrete decision on time duration of an entire stage. Moreover, the time period of an entire stage is available if and only if the stage ends. It could lead to unsafe driving behaviours. For example, when the DRL changes a green traffic light to a red one, drivers might be at the middle of crossroads and have high potentials of traffic accident. A rational traffic light control should determine the duration of each phase at the very beginning of the phase and drivers could be altered before the end of a phase.

Second, either the hand-crafted features from the environment [1, 4, 7] or the CNN-learned features [6, 9] via a snapshot of the vehicle positions nearby the intersections only capture static features, but not the inherently dynamics of traffic flows. Thus, with such static feature representation, DRL cannot precisely estimate traffic condition in a next phase and then make an inaccurate decision.

To tackle the issues above, in this paper, we propose a smarter and safer DRL-based traffic light control framework to learn a continuous time duration of traffic signal per phase, instead of making discrete actions. We expect that, beyond smooth traffic flow, our framework can lead to safer driving behaviour by introducing a transition phase before the changes of current phases. Such a transition phase can be intuitively treated as a yellow traffic light. As a summary, we make the following contribution in this paper.

- Our framework exploits the DDPG algorithm to learn a continuous time duration of traffic signal phases by a better DRL with more proper representation of state, action and reward.
- Our work introduces emergency stops as a part of reward and yellow phase into action space to optimize driving safety.
- Our initial evaluation on the popular traffic simulator SUMO validates the effectiveness and robustness of our method.

2 RELATED WORKS AND BACKGROUND

Intelligent Traffic Signal Control: Various deep learning and traffic data analytic techniques have been applied to solve traffic problems [3, 8, 10]. Recently, reinforcement learning algorithm has been used to solve complex traffic management problems, e.g., intelligent traffic light control via deep reinforcement learning techniques [1, 4–6, 9]. For example, the previous works [1, 4, 5, 9] make discrete decisions, where action space is a set of discrete actions containing all possible traffic signal phases. The DRL agent then decides a certain phase among such a set. The discrete decision-making suffers from unsafe driving because drivers have no idea of the change of red/green lights and are not even alerted of the remaining time of current phase. Also, due to discrete actions, the time period of an entire stage is essentially learned at regular intervals, and the optimal one is thus missed.

In addition, the work [2] takes into account all traffic lights within an intersection, and assumes that 1) the total time period for such all traffic lights is fixed by a predefined value, and 2) the change order of all such traffic lights is also fixed. Then among the predefined value, this work learns a time period for each phase of such a traffic light via a policy-based reinforcement learning method. Due to such two assumptions, this work still suffers from the following issues: even if the road from west to east does not have any traffic flow, the work [2] could still assign a certain number of time period for the traffic lights in this road.

Review of DDPG: Deep Deterministic Policy Gradient (DDPG) is an Actor-Critic reinforcement learning algorithm which learns both an Actor-network and a Critic-network. The Actor-network takes the state as input and generates continuous actions, while Critic-network is used to evaluate the goodness of an action in a certain state. Based on the output of Critic-network, the Actor-network then updates its parameter and Critic-network next updates its parameter by the reward obtained from environment.

3 SOLUTION DETAIL

In this section, we first formulate the problem definition and next give the solution detail.

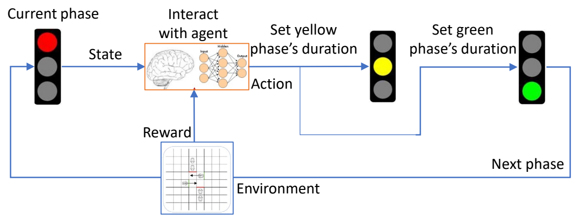

Figure 1: Framework of Traffic Signal Control

3.1 Problem Definition

Without loss of generality, as shown in Figure 1, we represent the environment by an intersection with two road segments. We denote four road directions, i.e., north, south, west and east by N, S, W and E, respectively, and have the two intersected road segments: the road N-S and one W-E. The traffic signal is with three phase: green (G), red (R) and yellow (Y). The time duration of each phase is dynamically learned according to real traffic condition. When the remaining time of current phase is 0, the agent will turn the traffic signal into yellow for a certain number of seconds, and meanwhile decide the time duration of next phase. Here, the yellow phase can be treated as a transit between the current and next phases for driving safety. Typically, from the current red phase to the next green phase, the time period of yellow phase is zero, and otherwise a small positive number. Depending upon the environment, the goal of the agent is to decide the transition time of the yellow phase meanwhile and the time duration of next phase. The time duration is learned by current traffic condition. We expect that the introduced yellow phase and learned continuous time duration make traffic flow become smoother and safer (i.e., positive reward).

3.2 Solution Detail

Our approach is based on DDGP. We first introduce the main components of DDGP including state, action and reward, and then outline the DDGP-based traffic signal control algorithm (Alg. 1).

3.2.1 State space. Given an intersection of two road segments (i.e., environment), our state includes different-grained time-spatial components. The coarse-grained components include waiting queue length, average delay of each lane, and waiting time of each car. Instead, the fine-grained component is a snapshot of all vehicles. In Figure 2, we divide the road close to the intersection into 3×15 cells. In our snapshot, each cell is with a pair of numbers, where the left number indicates whether there is a vehicle within the cell and the right one is the moving speed of this vehicle, if any. For simplicity, we do not plot the pairs for those empty cells without vehicles (the left numbers in such pairs are zeros). Note that our snapshot differs from the previous one [6, 9] who only takes vehicle positions as input with the left numbers alone. The purpose of introducing moving speed is as follows.

First, as a widely used parameter in the community of traffic control, *vehicle arrival rate* close to an intersection heavily depends upon moving speed. Here, the arrival rate refers to the count of vehicles arriving at the intersection every second. For example, in Figure 2, the moving speed of each vehicle in the left-most four columns of case A is 1 cell per second, and the one in the leftmost two columns of case B is 2 cells per second. Nevertheless, the arrival rates of both cases are 1.5 vehicles per second, which can more precisely explain traffic flow to the intersection. As a result, with help of the introduced moving speed, our snapshot can more precisely represent the traffic condition (i.e., environment).

Second, the left numbers in each non-empty pair above alone simply indicate the static location of every vehicle, and cannot capture dynamic traffic flow of moving vehicles. Without such dynamic flow, the agent cannot make safe decision. For example, when a vehicle is moving very fast to an intersection, it is more reasonable for the agent to adopt a relative longer transition period for yellow phase before the next red phase starts. Thus, we expect that the introduced speed, together with vehicle locations, could more precisely represent the traffic flow. The agent could make better decision for smooth traffic flow and driving safety.

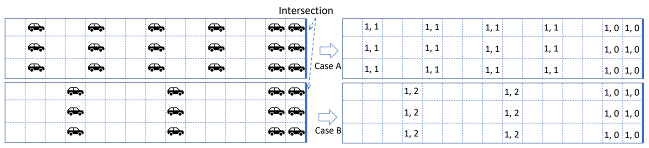

Figure 2: Two cases are with the same arrival rate to the intersection, though vehicle moving speed in the cases differs.

3.2.2 *Action space.* In our DDPG-based framework, the agent decides an action by a pair of continuous numbers $<t_{yellow},t_{next}>$, where t_{yellow} is the time duration of yellow phase and t_{next} the time duration of next phase. Recall that yellow phase is for driving safety and efficiency. Thus, the agent decides the number t_{yellow} in a flexible manner. That is, when vehicles with high speed enter the intersection, the agent could tune a relatively greater number t_{yellow}. The driver then could have longer time to drive across the intersection and avoid emergency braking when the red phase starts. In terms of t_{next}, the agent learns the time duration of next phase mainly for smooth traffic flow.

It is not hard to find that $<t_{yellow},t_{next}>$ could be better in line with the realistic setting and make safer driving. For example, after a pair of $<t_{yellow},t_{next}>$ is decided by the DDPG agent, we could highlight the remaining counters (time period) in traffic control lights to alert drivers for safer and more smooth driving.

3.2.3 *Reward.* Since our goal of traffic light control is for more efficient and safer driving. In Equation 1, we design the reward as the weighted sum of the following factors.

- D: sum of delay for all vehicles entering the intersection.
- L: queue length of all lanes entering the intersection.
- W: sum of waiting times for all vehicles in the entering lanes of intersection.
- E: sum of emergency braking in one phase. We empirically define emergency braking when the average vehicle deceleration is greater than $4.5 m/s^2$.
- N: number of vehicles passing through the intersection in this phase.
- T: average travel time of all vehicles passing through the intersection in this phase.
- We empirically set $w_1 = w_3 = w_4 = w_6 = -0.25$, $w_2 = -5$, and $w_5 = 1$.

$$Reward = w_1 * D + w_2 * L + w_3 * W + w_4 * E + w_5 * N + w_6 * T \quad (1)$$

3.2.4 *Algorithm Detail.* Alg. 1 first initializes the parameters of four networks: critic-actor networks, target critic-actor networks (lines 1-2). After that, within the two **for** loops, the first stage (lines 7-10) leverages the sampled actions generated by actor network α_t to interact with environment and the collected experiences is inserted into a replay buffer R. In the second stage (lines 11-15), when given sufficient experience, we train the actor and critic networks Q and θ by the experience in replay buffer. After that, we softly update the target networks $\theta^{\pi'}$ and $\theta^{Q'}$.

Algorithm 1 DDPG-based Traffic Signal Control

```
1:  Randomly initialize the critic and actor networks: $\theta^Q$ , $\theta^\pi$;
2:  Initialize the target critic and actor networks: $\theta^{Q'}$=$\theta^Q$, $\theta^{\pi'}$=$\theta^\pi$;
3:  Initialize replay buffer $R$;
4:  for $m$=1 to max-iterations do
5:      Reset environment, get initial state $s_0$;
6:      for $t$=0 to $T$ do
        ▷ Stage 1: collect experiences
7:          Sample actions $a_t = \pi(s)$;
8:          Execute $a_t$ in simulator
9:          observe reward $r_t$ and next state $s_t + 1$
10:         Store transition $(s_t, a_t, r, s_t + 1)$ in R
        ▷ Stage 2: train networks
11:         Randomly sample $N$ transitions $(s_t, a_t, r, s_t + 1)$ from $R$
12:         Set $y_n = r_n + Q'(s_{n+1}, \pi'(s_n + 1))$
13:         Update critic Q to minimize $\sum_n (y_n - Q(s_n, a_n)^2)$
14:         Update actor $\pi$ to maximize $\sum_n Q(s_n, \pi(s_n))$
15:         Upt target $\theta^{\pi'} \leftarrow \tau\theta^\pi + (1-\tau)\theta^{\pi'}, \theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
```

4 EXPERIMENT

In this section, we perform preliminary experiments to evaluate the performance of the DDPG-based traffic signal control by a well-known traffic simulator Urban Mobility (SUMO). It simulates traffic on large road networks. Table 5 lists three configurations of traffic data in our experiments. All these configurations are four-way intersection with different arrival rate, speed limitation and traffic density. In Table 5, the arrival rate refers to the number of vehicles entering the intersection per second, and the arrival vehicles are generated by Poisson distribution with certain arrival rates. There are various speed limitations for roads in every direction. For example, in Configuration #2, the speed limitation of two west-east and south-north roads is 19.44 m/s and 8.33 m/s, respectively. Lastly, the start and end time leads to various traffic density.

4.1 Experiment settings

The environment is conducted on one four-way intersection. Each road has six lanes: three for entering and three for leaving the intersection. Table 2 lists the mainly used parameters. The traffic signal on the intersection has three phases:

- Green light on W-E direction, red light on N-S direction.
- Red light on W-E direction, green light on N-S direction.
- Yellow light on all direction.

Configuration	Directions	Arrival rate (cars/s)	Speed limitation (m/s)	Start time (s)	End time (s)
1	W-E/S-N	0.4/0.4	19.44/19.44	0/36001	36000/72000
2	W-E/S-N	0.4/0.4	19.44/8.33	0/0	72000/72000
3	W-E/S-N	0.4/0.01	19.44/19.44	0/0	72000/72000

Table 1: Configurations for traffic data

Model parameters	Value
Duration range	0-180 seconds
Yellow phase	0-5 seconds
Replay size	500
Batch size	64
Discount factor γ	0.9
Exploration factor ϵ	0.05
Learning rate α	0.001

Table 2: Parameters for experiment

The intersection above, though rather simple, could evaluate the advantage of our work. We believe that our work still works on more complex intersections and real-world traffic in future work.

4.2 Performance comparison

We compare our work against two alternative approaches. (1) FIX: this approach adopts a fixed time duration 30 seconds for each phase. (2) DQN: this approach uses the DQN algorithm for discrete action-based traffic light control. Action is taken every 5 seconds, and is defined as a = 1: change the light to next phase, and a = 0: keep the current phase. (3) DDPG: our DDPG-based framework.

Method	Queue length	Waiting time	Delay	Emergency stops
FIX	8.049	2.472	1.457	7
DQN	0.302	2.449	0.024	6
DDPG	0.014	1.645	0.009	2

Table 3: Performance on Configuration #1

Method	Queue length	Waiting time	Delay	Emergency stops
FIX	8.869	2.739	1.623	6
DQN	4.736	3.051	0.727	7
DDPG	0.034	1.992	0.038	1

Table 4: Performance on Configuration #2

Method	Queue length	Waiting time	Delay	Emergency stops
FIX	4.432	1.838	3.459	7
DQN	0.671	1.301	2.154	10
DDPG	0.386	0.324	2.050	2

Table 5: Performance on Configuration #3

Tables 3, 4 and 5 give the evaluation results on three configurations. First, we can find that that the yellow phase in our work does help a significantly lower number of emergency stops in all configurations. Such a number leads to safer driving. In addition, the introduction of vehicle speed into state leads to much higher efficiency particularly for Configurations #1 and #2. Furthermore, our approach learns continuous time duration for each phase, leading to more flexible control to traffic signal, when compared with fixed-time control or discrete control.

5 CONCLUSION

In this paper, we propose a DDPG-based traffic signal control to decide a continuous time duration for each phase. By introducing vehicle speed into state and yellow transit phase into action space, the proposed DDPG decision can lead to more efficient traffic flow and safer driving behaviour. The preliminary experiment on SUMO validates that our work performs significantly better than the state-of-the-art DQN-based decision in terms much smaller number of emergency stops, shorter queue length, smaller waiting time and trivial delay.

As the future work, we plan to 1) extend our work from a single intersection to a large-scale road network with much more intersections, and 2) perform the evaluation on real world dataset with much complex traffic patterns.

ACKNOWLEDGE

This work is partially supported by National Natural Science Foundation of China (Grant No. 61972286).

REFERENCES

[1] M. Abdoos, N. Mozayani, and A. L. Bazzan. Holonic multi-agent system for traffic signals control. *Engineering Applications of Artificial Intelligence*, 26(5-6):1575–1587, 2013.
[2] N. Casas. Deep deterministic policy gradient for urban traffic light control. *arXiv preprint arXiv:1703.09035*, 2017.
[3] X. Di, Y. Xiao, C. Zhu, Y. Deng, Q. Zhao, and W. Rao. Traffic congestion prediction by spatiotemporal propagation patterns. In *20th IEEE MDM*, pages 298–303, 2019.
[4] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad. Design of reinforcement learning parameters for seamless application of adaptive traffic signal control. *Journal of Intelligent Transportation Systems*, 18(3):227–245, 2014.
[5] L. Li, Y. Lv, and F.-Y. Wang. Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, 3(3):247–254, 2016.
[6] X. Liang, X. Du, G. Wang, and Z. Han. A deep reinforcement learning network for traffic light cycle control. *IEEE Transactions on Vehicular Technology*, 68(2):1243–1253, 2019.
[7] S. S. Mousavi, M. Schukat, and E. Howley. Traffic light control using deep policy-gradient and value-function-based reinforcement learning. *IET Intelligent Transport Systems*, 11(7):417–423, 2017.
[8] D. Srinivasan, M. C. Choy, and R. L. Cheu. Neural networks for real-time traffic signal control. *IEEE Transactions on intelligent transportation systems*, 7(3):261–272, 2006.
[9] H. Wei, G. Zheng, H. Yao, and Z. Li. Intellilight: A reinforcement learning approach for intelligent traffic light control. In *Proceedings of the 24th ACM SIGKDD*, pages 2496–2505, 2018.
[10] W. Wei and Y. Zhang. Fl-fn based traffic signal control. In *2002 IEEE World Congress on Computational Intelligence. 2002 IEEE International Conference on Fuzzy Systems. FUZZ-IEEE'02. Proceedings (Cat. No. 02CH37291)*, volume 1, pages 296–300. IEEE, 2002.