IntelliLight: a Reinforcement Learning Approach for Intelligent Traffic Light Control

Hua Wei, Guanjie Zheng, Huaxiu Yao, Zhenhui Li*
The Pennsylvania State University
University Park, PA
hzw77,gjz5038,huaxiuyao,jessieli@ist.psu.edu

ABSTRACT

The intelligent traffic light control is critical for an efficient transportation system. While existing traffic lights are mostly operated by hand-crafted rules, an intelligent traffic light control system should be dynamically adjusted to real-time traffic. There is an emerging trend of using deep reinforcement learning technique for traffic light control and recent studies have shown promising results. However, existing studies have not yet tested the methods on the real-world traffic data and they only focus on studying the rewards without interpreting the policies. In this paper, we propose a more effective deep reinforcement learning model for traffic light control. We test our method on a large-scale real traffic dataset obtained from surveillance cameras. We also show some interesting case studies of policies learned from the real data.

CCS CONCEPTS

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

KEYWORDS

Traffic light control; reinforcement learning

1 INTRODUCTION

Traffic congestion has become increasingly costly. For example, traffic congestion costs Americans $124 billion a year, according to a report by Forbes in 2014 [12]. In European Union, the traffic congestion cost is estimated to be 1% of its GDP [7]. Improving traffic conditions could increase city efficiency, improve economy, and ease people's daily life.

One way to reduce the traffic congestion is by intelligently controlling traffic lights. Nowadays, most traffic lights are still controlled with pre-defined fixed-time plan [18, 23] and are not designed by observing real traffic. Recent studies propose hand-crafted rules according to real traffic data [5, 20]. However, these rules are still pre-defined and cannot be dynamically adjusted w.r.t. real-time traffic.

*The first two authors Hua Wei and Guanjie Zheng contribute equally to this paper.

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220096

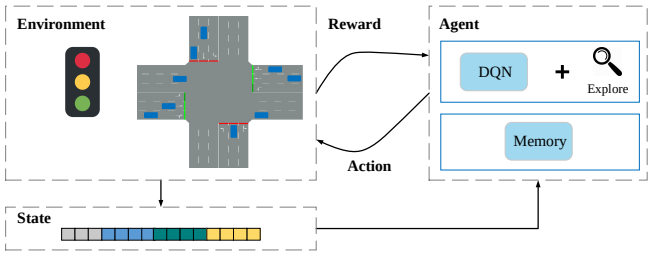


Figure 1: Deep reinforcement learning framework for traffic light control.

To dynamically adjust traffic lights according to real-time traffic, people have been using reinforcement learning technique [13, 22, 24]. Traditional reinforcement learning is difficult to apply due to two key challenges: (1) how to represent environment; and (2) how to model the correlation between environment and decision. To address these two challenges, recent studies [15, 22] have applied deep reinforcement learning techniques, such as Deep Q-learning (DQN), for traffic light control problem. Figure 1 illustrates the basic idea of deep reinforcement learning framework. The environment is composed of traffic light phase and traffic condition. State is a feature representation of the environment. Agent takes state as input and learns a model to predict whether to "keep the current phase of traffic lights" or "change the current phase". 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 timestamp based on the new state and the updated model. In such a framework, traffic condition can be described as an image and such an image is directly taken as an input for a CNN-based model to enrich the hand-crafted features of the environment.

Recent deep reinforcement learning approaches made promising progress for the traffic light control problem. Our approach extends this line of work by making several important improvements, particularly with the emphasize of studies using real-world data. Our key contributions are as follows.

1. Experiments with real traffic data. Nowadays, increasing amount of traffic data is being collected from various sources. In China, many big cities have installed AI-equipped traffic surveillance cameras on roadside to monitor traffic conditions in real time. Such real-time traffic data enables us to implement reinforcement learning in real world. However, to the best of our knowledge, none of existing studies have used the real traffic data to test their methods. Instead, they use traffic simulations and such simulations

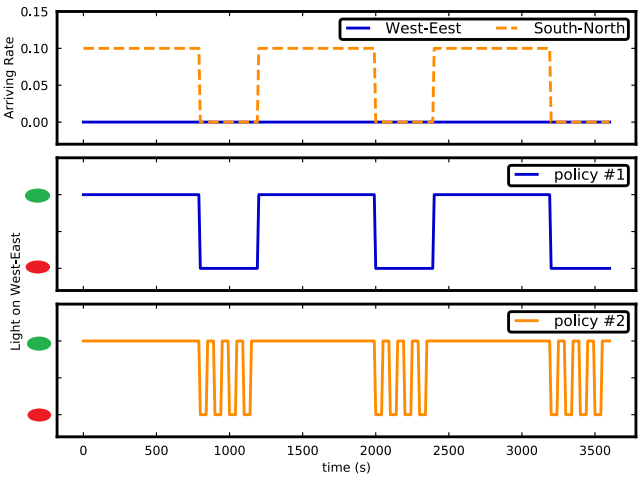


Figure 2: Reward is not a comprehensive measure to evaluate traffic light control performance. Both policies will lead to the same rewards. But policy #1 is more suitable than policy #2 in the real world.

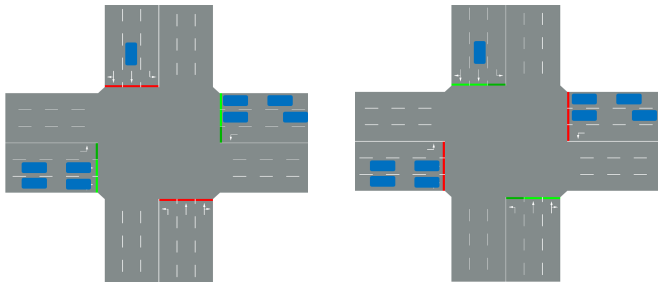


Figure 3: Case A and case B have the same environment except the traffic light phase.

do not reflect the real-world traffic. For example, the simulation models in current studies often assume that vehicles arrive at a constant rate but real traffic are highly dynamic over time. In our paper, we test the methods on a large-scale real traffic data obtained from 1,704 surveillance cameras in Jinan, China for a period of 31 days (see experiment section for details). In this dataset, there are more than 405 million vehicle records and more than 11 million unique vehicle plates. We conduct comprehensive experiments on such large real dataset.

2. Interpretations of the policy. A frequently used measure to quantify the performance of traffic light control is by examining the overall reward, which can be defined by several factors such as waiting time of vehicles and number of vehicles passing the intersections. However, existing studies rarely make observations of the policy learned from the model. The reward could be misleading in some cases. There could be different policies with the same reward but one is more suitable than the other. Take Figure 2 as an example. Assume there is only traffic on South-North direction and the traffic comes for the first 800 seconds in every 1200 seconds. Policy #1 is 800 seconds for green light on South-North direction and followed by red light for 400 seconds, and then repeat. And policy #2 is different from policy #1 in the way that, instead of 400-second red light on South-North direction, the light changes every 100 seconds. Both policies will result in the same reward because no vehicle will be waiting under either policy. However, policy #1 is preferred over policy #2 in real scenario. In this paper, we claim that it is important to study the policies rather than simply showing the overall reward. In our experiments, we show several interesting policies learned from the real traffic under different scenarios (e.g., peak hours vs. non-peak hours, weekday vs. weekend).

3. A phase-gated model learning. As described earlier in deep reinforcement learning framework, the agent will take the state, which is the representation of environment, as model input. The environment usually includes the current traffic light phase and traffic conditions. For example, the environments of two cases in Figure 3 are the same, except the traffic light phases. Previous studies all take phase as one feature [17, 22], together with many other features (e.g., number of vehicles at different lanes, positions of vehicles). And it is likely that this one feature does not play a role that is significant enough to affect the model output. Therefore, the model will make the same decision (i.e., either keep or change the current phase) for these two different cases. However, such a decision, no matter which one, is not ideal for one of the cases. Because in Figure 3, case A hopes to keep the phase and case B hopes to change the phase. In this paper, we propose a new phase-sensitive (i.e., phase gate combined with memory palace) reinforcement learning agent, which is a critical component that leads to superior performance.

The rest of this paper is organized as follows. In Section 2, we review the literature. Then, in Section 3, we define the problem. The method is shown in Section 4 and the experimental results on synthetic data and real data are shown in Section 5. Finally, we conclude the paper in Section 6.

2 RELATED WORK

In this section, we firstly introduce conventional methods for traffic light control, then introduce methods using reinforcement learning.

2.1 Conventional Traffic Light Control

Early traffic light control methods can be roughly classified into two groups. The first is pre-timed signal control [6, 18, 23], where a fixed time is determined for all green phases according to historical traffic demand, without considering possible fluctuations in traffic demand. The second is vehicle-actuated control methods [5, 20] where the real-time traffic information is used. Vehicle-actuated methods are suitable for the situations with relatively high traffic randomness. However, this method largely depends on the handcraft rules for current traffic condition, without taking into account future situation. Therefore, it cannot reach the global optimal.

2.2 Reinforcement Learning for Traffic Light Control

Recently, due to the incapability of dealing with dynamic multi-direction traffic in previous methods, more works try to use reinforcement learning algorithms to solve the traffic light control problem [13, 17, 24]. Typically, these algorithms take the traffic on the road as state, and the operation on light as action. These methods usually show better performance compared with fixed-time and traffic-responsive control methods.

Methods in [1, 2, 4, 8, 24] designed the state as discrete values like the location of vehicles or number of waited cars. However, the discrete state-action pair value matrix requires huge storage space, which keeps these methods from being used in large state space problems.

To solve the in-managablely large state space of previous methods, recent studies [15, 22] propose to apply Deep Q-learning methods using continuous state representations. These studies learn a Q-function (e.g. a deep neural network) to map state and action to reward. These works vary in the state representation including hand craft features (e.g., queue length [15, 17], average delay [10, 22]) and image features[9, 16, 22]) They are also different in reward design, including average delay [3, 22],the average travel time [16, 22], and queue length[15].

However, all these methods assume relatively static traffic environments, and hence far from the real case. Further, they only focus on rewards and overlook the adaptability of the algorithms to the real traffic. Therefore, they cannot interpret why the learned light signal changes corresponding to the traffic. In this paper, we try to test the algorithms in a more realistic traffic setting, and add more interpretation other than reward.

3 PROBLEM DEFINITION

In our problem, we have the environment E as an intersection of two roads (and the traffic on this intersection). There is an intelligent traffic light agent G. To make the notation simpler, we use "N", "S", "W", "E" to represent north, south, west, and east respectively, and use "Red' and "Green" to represent red light and green light correspondingly. A setting of the traffic light is defined as a phase (e.g., green light on the west-east direction which can be simplified as Green-WE). When a light changes from green to red, there is a 3 second yellow light, while the other directions still keep red. So one green light and the subsequent yellow light can be represented together by "Green". To simplify the problem, we assume there is only two phases of the traffic light, i.e., 1) Green-WE, and 2) Red-WE. Due to the limitation of real-world setting, the traffic light can only change in a specific order (i.e., 1 -> 2 -> 1 -> 2 -> ...). Given the state s (describing the positions and speed of the traffic near this intersection), the goal of the agent G is to give the optimal action a (i.e., whether to change the light to the next phase), so that the reward r (i.e., the smoothness of traffic) can be maximized.

4 METHOD

Traffic light control has attracted a lot of attention in recent years due to its essential role in adjusting traffic. Current methods generally have two categories, conventional methods, and deep reinforcement learning based methods. Conventional methods usually rely on previous knowledge to set fixed time for each light phase or set changing rules. These rules are prone to dynamically changing traffic. Reinforcement learning methods usually take the traffic condition (e.g., queue length of waiting cars and average waiting time) as state, and try to make actions that can improve the traffic condition based on the current state.

However, the current methods do not consider the complex situations in real case, and hence may lead to stuck in one single kind of action. This will lead to inferior traffic adjusting performance under complex traffic situation.

In this section, we propose a deep reinforcement traffic light agent to solve this problem. We will first introduce the model framework in Section 4.1. Then, we show the design of agent in Section 4.2. We further describe the network structure in Section 4.3. In addition, we describe the memory palace in Section 4.4. Note that, although our model is designed for a four way intersection with two phases, it is not difficult to extend it to other types of intersections or to multiple phases scenarios.

Table 1: Notations

Notation	Meaning
E	Environment
G	Agent
a	Action
s	State
r	Reward
Δt	Time interval between actions
q	Action value function
Q	Deep Q-Network
$\mathsf{L_i}$	Queue length on lane i
$\mathsf{V_i}$	Number of vehicles on lane i
$\mathsf{W_i}$	Sum of waiting time of vehicles on lane i
$\mathsf{M} \in \mathbb{R}^{N \times N}$	Image representation of vehicles' position
P_c	Current phase
P_n	Next phase
$\mathsf{C} \in \{0, 1\}$	Light switches (1) or not (0)
$\mathsf{D_j}$	Delay of vehicle j.
N	Number of vehicles passed the intersection after the action.
T	Sum of the travel time in system of vehicles passed the intersection after action a .

4.1 Framework

Our model is composed of offline part and online part (as shown in Figure 4). We extract five kinds of features describing the traffic conditions as state (detailed in Section 4.2), and use reward to describe how much the action has improved the traffic (detailed in Section 4.2). In offline stage, we set a fixed timetable for the lights, and let traffic go through the system to collect data samples. After training with the samples logged in this stage, the model will be put into the online part. In online stage, at every time interval Δt, the traffic light agent will observe the state s from the environment and take action a (i.e., whether to change light signal to the next phase) according to ϵ-greedy strategy combining exploration (i.e., random

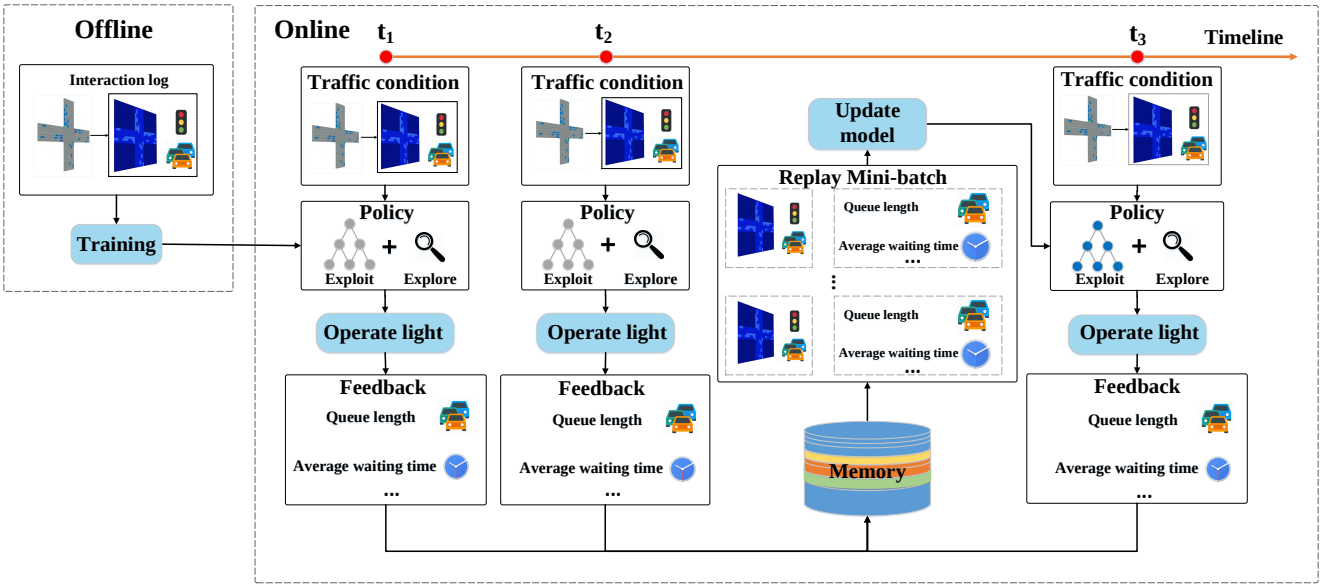


Figure 4: Model framework

action with probability ϵ) and exploitation (i.e., the estimation of the potential reward of doing this action given state s). After that, the agent G will observe the environment and get the reward r from it. Then, the tuple (s, a, r) will be stored into memory. After several timestamps (e.g., t_2 in Figure 4), agent G will update the network according to the logs in the memory.

4.2 Agent Design

First, we introduce the state, action and reward representation.

- **State.** Our state representation includes queue length on lane i L_i, number of vehicles on lane i V_i, average waiting time W_i of vehicles on lane i, an image representation of vehicles' position M, current phase P_c and next phase P_n.
- **Action.** Our action is defined as a = 1: change the light to next phase P_n, and a = 0: keep the current phase P_c.
- **Reward.** As is shown in Equation 1, reward is defined as a weighted sum of the following factors:

(1) Sum of queue length L for lanes, where L is calculated as the total number of halting vehicles for the last time step on the given lane. A speed of less than 0.1 m/s is considered a halt.

(2) Sum of average waiting time W for lanes, where W is defined as the time (in minutes) a vehicle spent with a speed below 0.1m/s since the last time it was faster than 0.1m/s. Basically, the waiting time of a vehicle is reset to 0 every time it moves.

(3) Number of light switches C, where C = 0 for keeping the light, and 1 for changing the light.

(4) Sum of delay D for vehicles as $1 - \frac{vehicle\_speed}{allowed\_speed}$.

(5) Total number of vehicles that passed the intersection during the time interval after action a.

(6) Sum of the travel time of vehicles that passed the intersection during the time interval after action a, defined as the total time (in minutes) that vehicles spent on approaching lanes.

$$Reward = w_1 * \sum_{i \in I} L_i + w_2 * \sum_{i \in I} W_i + w_3 * C + w_4 * \sum_{j \in J} D_j + w_5 * N + w_6 * T. \quad (1)$$

Hence, given the current state s of the traffic condition, the mission of the agent G is to find the action a (change or keep light) that may lead to the maximum reward r in the long run, following the Bellman Equation (Equation 2) [21]. In this situation, the action value function of the current situation is the summation of the reward of the right next step and maximum potential future reward. Through this conjecture of future, the agent can select action that is more suitable for long-run reward.

$$q(s_t, a, t) = r_{a,t+1} + \gamma \max q(s_{a,t+1}, a', t+1) \quad (2)$$

4.3 Network Structure

In order to estimate the reward based on the state, and action, the agent needs to learn a Deep Q-Network Q(s, a).

In the real-world scenario, traffic is very complex and contain many different cases need to be considered separately. We will illustrate this in Example 4.1.

Example 4.1. We still assume a simple intersection with two-phase light transition here: 1) Green-WE, and 2) Red-WE. The decision process of whether to change the traffic light consists of two steps. The first step is the mapping from traffic condition (e.g., how many cars are waiting, how long has each car been waiting) to a partial reward. An example of this mapping could be

$r = -0.5 \times L - 0.7 \times W$. This is shared by different phases, no matter which lane the green light is on. Then, to determine the action, the agent should watch on the traffic in different lanes during different phases. For instance, as is shown in Figure 3 (a), when the red light is on the *NS* direction, more waiting traffic (i.e., lower reward in the first step) on the *NS* direction will make the light tend to change (because by changing the light on this lane from red to green, more cars on this lane can pass through this intersection), while more waiting traffic (i.e., lower reward in the first step) on the *WE* direction will make the light tend to keep. When the red light is on the *WE* direction, the case is right the opposite. Therefore, the light phase should have an explicit selection on features.

In previous studies, due to the simplified design of the model for approximating Q-function under complex traffic condition, agents are having difficulties in distinguishing the decision process for different phases. Therefore, we hereby propose a network structure that can explicitly consider the different cases explicitly. We call this special sub-structure "Phase Gate".

Our whole network structure can be shown as in Figure 5. The image features are extracted from the observations of the traffic condition and feed into two convolutional layers. The output of these layers are concatenated with the four explicitly mined features, queue length L, average waiting time W, phase P and number of total vehicles V. The concatenated features are then fed into fully-connected layers to learn the mapping from traffic conditions to potential rewards. Then, for each phase, we design a separate learning process of mapping from rewards to the value of making decisions Q(s, a). These separate processes are selected through a gate controlled by the phase. As shown in Figure 5, when phase $P = 0$, the left branch will be activated, while when phase $P = 1$, the right branch will be activated. This will distinguish the decision process for different phases, prevent their decision from favoring certain action, and enhance the fitting ability of the network.

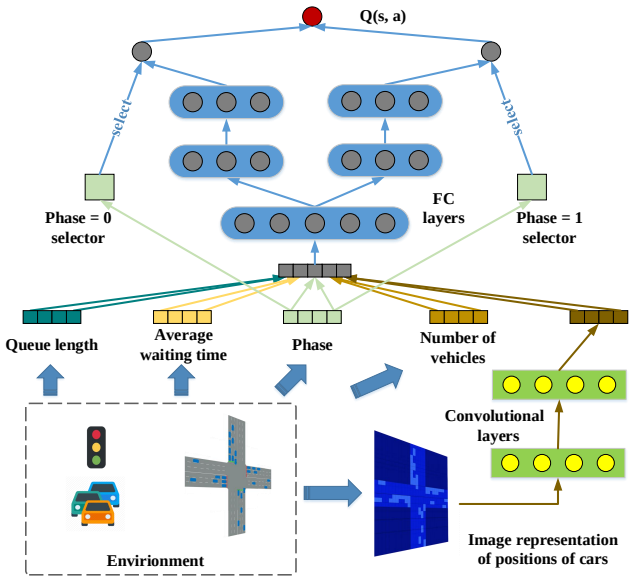


Figure 5: Q-network

4.4 Memory Palace and Model Updating

Periodically, the agent will take samples from the memory and use them to update the network. This memory is maintained by adding the new data samples in and removing the old samples occasionally. This technique is noted as experience replay [19] and has been widely used in reinforcement learning models.

However, in the real traffic setting, traffic on different lanes can be really imbalanced. As previous methods [9, 10, 15, 22] store all the state-action-reward training samples in one memory, this memory will be dominated by the phases and actions that appear most frequently in imbalanced settings. Then, the agent will be learned to estimate the reward for these frequent phase-action combinations well, but ignore other less frequent phase-action combinations. This will cause the learned agent to make bad decisions on the infrequent phase-action combinations. Therefore, when traffic on different lanes are dramatically different, these imbalanced samples will lead to inferior performance on less frequent situation.

Inspired by Memory Palace theory [11, 14] in cognitive psychology, we can solve this imbalance by using different memory palaces for different phase-action combinations. As shown in Figure 6, training samples for different phase-action combinations are stored into

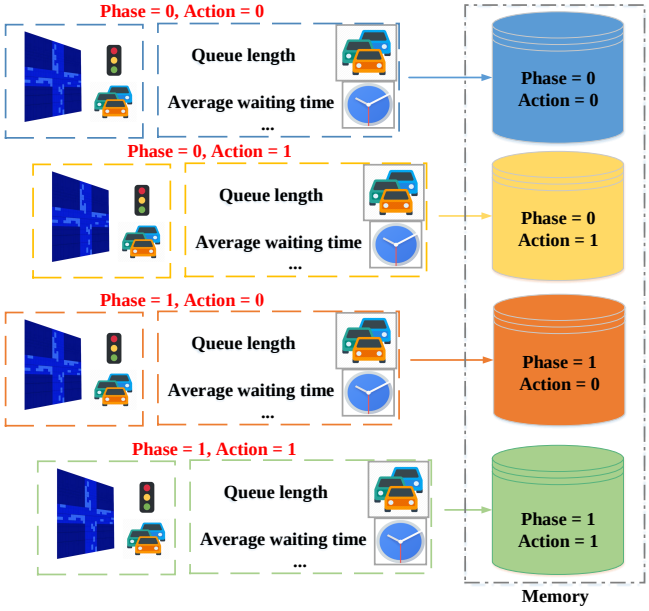


Figure 6: Memory palace structure

different memory palaces. Then same number of samples will be selected from different palaces. These balanced samples will prevent different phase-action combinations from interfering each other's training process, and hence, improve the fitting capability of the network to predict the reward accurately.

5 EXPERIMENT

In this section, we conduct experiments using both synthetic and real-world traffic data. We show a comprehensive quantitative evaluation by comparing with other methods and also show some interesting case studies [1].

5.1 Experiment Setting

The experiments are conducted on a simulation platform SUMO (Simulation of Urban MObility) [2]. SUMO provides flexible APIs for road network design, traffic volume simulation and traffic light control. Specifically, SUMO can control the traffic moving according to the given policy of traffic light (obtained by the traffic light agent).

The environment for the experiments on synthetic data is a four-way intersection as Figure 2. The intersection is connected with four road segments of 150-meters long, where each road have three incoming and three outgoing lanes. The traffic light in this part of experiment contains two phases: (1) Green-WE (green light on *WE* with red light on *SN*), (2) Red-WE (red light on *WE* with green light on *SN*). Note that when a green light is on one direction, there is a red light on the other direction. Also, a green light is followed by a 3-second yellow light before it turns to red light. Although this is a simplification of the real world scenario, the research of more types of intersections (e.g., three-way intersection), and more complex light phasing (e.g., with left-turn phasing) can be further conducted in similar way.

5.2 Parameter Setting

The parameter setting and reward coefficient for our methods is shown in Table 2 and Table 3 respectively. We found out that the action time interval Δt has minimal influence on performance of our model as long as Δt is between 5 seconds to 25 seconds.

Table 2: Settings for our method

Model parameter	Value
Action time interval Δt	5 seconds
γ for future reward	0.8
ϵ for exploration	0.05
Sample size	300
Memory length	1000
Model update interval	300 seconds

5.3 Evaluation Metric

We evaluate the performance of different methods on five measures: 1) average reward defined as Equation 1 for every second, 2) average queue length (sum of the number of waiting cars) for every second, 3) average delay for every second, 4) average waiting time (in minutes) of all lanes for every second, and 5) average duration in seconds (the average travel time that every vehicle spends on approaching lanes) for every vehicle. A higher reward indicates a better performance of the method. A smaller queue length, delay, waiting time and duration indicates the traffic is less jammed. Note that, the reward is the combination of several terms (positive and negative terms), therefore, the range of reward is from $-\infty$ to ∞. (Under specific configuration, there will be an upper bound for the reward when all cars move freely without any stop or delay.)

[1] Codes are available at the author's website.
[2] http://sumo.dlr.de/index.html

Table 3: Reward coefficient

Reward coefficient	Value
w_1	-0.25
w_2	-0.25
w_3	-5
w_4	-0.25
w_5	1
w_6	1

5.4 Compared Methods

To evaluate the effectiveness of our model, we compare our model with the following baseline methods, and tune the parameter for all methods. We then report their best performance.

- **Fixed-time Control (*FT*)**. Fixed-time control method use a pre-determined cycle and phase time plan [18] and is widely used when the traffic flow is steady.
- **Self-Organizing Traffic Light Control (*SOTL*)** [5]. This method controls the traffic light according to the current traffic state, including the eclipsed time and the number of vehicles waiting at the red light. Specifically, the traffic light will change when the number of waiting cars is above a hand-tuned threshold.
- **Deep Reinforcement Learning for Traffic Light Control (*DRL*)**. Proposed in [22], this method applies DQN framework to select optimal light configurations for traffic intersections. Specifically, it solely relies on the original traffic information as an image.

In addition to the baseline methods, we also consider several variations of our model.

- ***IntelliLight* (*Base*)**. Using the same network structure and reward function defined as in Section 4.2 and 4.3. This method is without Memory Palace and Phase Gate.
- ***IntelliLight* (*Base+MP*)**. By adding Memory Palace in psychology to *IntelliLight-Base*, we store the samples from different phase and time in separate memories.
- ***IntelliLight* (*Base+MP+PG*)**. This is the model adding two techniques (Memory Palace and Phase Gate).

5.5 Datasets

5.5.1 Synthetic data. In the first part of our experiment, synthetic data is used with four traffic flow settings: simple changing traffic (configuration 1), equally steady traffic (configuration 2), unequally steady traffic (configuration 3) and complex traffic (configuration 4) which is a combination of previous three configurations. As is shown in Table 4, the arriving of vehicles are generated by Poisson distribution with certain arrival rates.

5.5.2 Real-world data. The real-world dataset is collected by 1,704 surveillance cameras in Jinan, China over the time period from

Table 4: Configurations for synthetic traffic data

Config	Directions	Arrival rate (cars/s)	Start time (s)	End time (s)
1	*WE*	0.4	0	36000
	SN	0.4	36001	72000
2	*WE*	0.033	0	72000
	SN	0.033	0	72000
3	*WE*	0.2	0	72000
	SN	0.033	0	72000
4	Configuration 1		0	72000
	Configuration 2		72001	144000
	Configuration 3		144001	216000

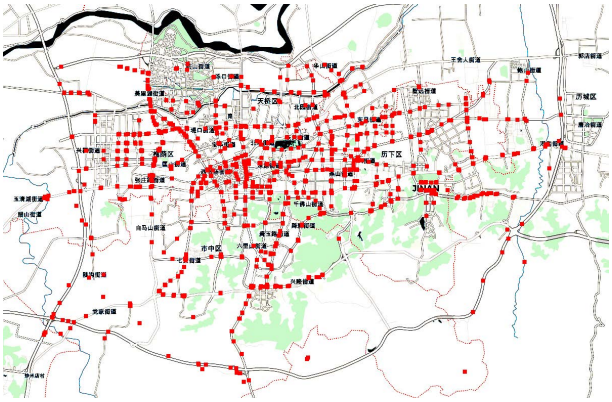


Figure 7: Traffic surveillance cameras in Jinan, China

08/01/2016 to 08/31/2016. The locations of these cameras are shown in Figure 7. Gathered every second by the cameras facing towards vehicles near intersections, each record in the dataset consists of time, camera ID and the information about vehicles. By analyzing these records with camera locations, the trajectories of vehicles are recorded when they pass through road intersections. The dataset covers 935 locations, where 43 of them are four-way intersections. We use the number of vehicles passing through 24 intersections as traffic volume for experiments since only these intersections have consecutive data. Then we feed this real-world traffic setting into SUMO as online experiments. It can be seen from Table 5 that traffic flow on different roads are dynamically changing in the real world.

Table 5: Details of real-world traffic dataset

Time Range	Records	Arrival Rate (cars/s)			
		Mean	Std	Max	Min
08/01/2016 - 08/31/2016	405,370,631	0.089	0.117	0.844	0.0

5.6 Performance on Synthetic Data

5.6.1 Comparison with state-of-the-art methods. We first compare our method with three other baselines under different synthetic traffic settings. From Table 6, 7, 8 and 9 we can see that our method performs better than all other baseline methods in configurations 1, 2, 3 and 4. Although some baselines perform well on certain setting, they perform badly in other configurations (e.g., *SOTL* achieves good rewards under configuration 1, almost the same as our method in 3 digit floats, this is because our method has learned to keep the light until 36000 s and switch the light after that, hence, these two methods perform very similar). On the contrary, our method *IntelliLight* shows steadily higher rewards under different configurations.

5.6.2 Comparison with variants of our proposed method. Table 6, 7, 8 and 9 show the performance of variants of our proposed method. First, we can see that adding Memory Palace helps achieve higher reward under configuration 3 and 4, although it does not boost the performance under configuration 1 and 2. This is because for the simple case (configuration 1 and 2), the phase is relatively steady for a long time (because the traffic only comes from one direction or keeps not changing in a long time). Therefore, the memory palace does not help in building a better model for predicting the reward. Further adding Phase Gate also reduces the waiting queue length in most cases and achieves highest reward, demonstrating the effectiveness of these two techniques.

Table 6: Performance on configuration 1. Reward: the higher the better. Other measures: the lower the better. Same with the following tables.

Model name	Reward	Queue length	Delay	Waiting time	Duration
FT	-2.304	8.532	2.479	1.619	42.230
SOTL	0.398	0.006	1.598	**0.001**	**24.129**
DRL	-36.247	91.412	4.483	62.521	277.430
IntelliLight (ours)					
Base	-3.077	10.654	2.635	4.047	92.080
Base+MP	-3.267	6.087	1.865	8.808	38.230
Base+MP+PG	**0.399**	**0.005**	**1.598**	**0.001**	**24.130**

Table 7: Performance on configuration 2

Model name	Reward	Queue length	Delay	Waiting time	Duration
FT	-0.978	1.105	2.614	0.216	34.278
SOTL	-21.952	19.874	4.384	65.406	177.747
DRL	-2.208	3.405	3.431	2.416	52.075
IntelliLight (ours)					
Base	-0.523	0.208	1.689	0.056	27.505
Base+MP	-0.556	0.259	1.730	0.085	27.888
Base+MP+PG	**-0.514**	**0.201**	**1.697**	**0.025**	**27.451**

Table 8: Performance on configuration 3

Model name	Reward	Queue length	Delay	Waiting time	Duration
FT	-1.724	4.159	3.551	0.797	36.893
SOTL	-20.680	20.227	5.277	60.427	69.838
DRL	-8.108	16.968	4.704	13.360	66.485
IntelliLight (ours)					
Base	-0.836	0.905	2.699	0.403	28.197
Base+MP	-0.698	0.606	2.729	0.128	26.948
Base+MP+PG	**-0.648**	**0.524**	**2.584**	**0.112**	**26.647**

Table 9: Performance on configuration 4

Model name	Reward	Queue length	Delay	Waiting time	Duration
FT	-1.670	4.601	2.883	0.878	39.707
SOTL	-14.079	13.372	3.753	41.949	54.014
DRL	-49.011	91.887	4.917	111.821	469.417
IntelliLight (ours)					
Base	-5.030	5.880	3.432	13.012	39.021
Base+MP	-3.329	5.358	2.238	7.164	44.703
Base+MP+PG	**-0.474**	**0.548**	**2.202**	**0.431**	**25.977**

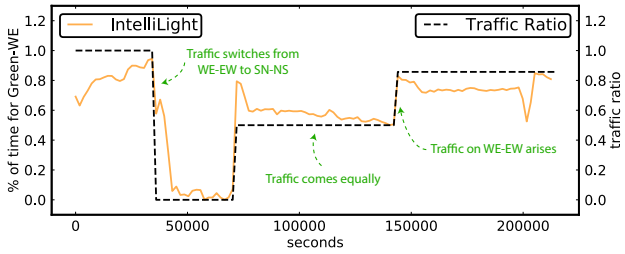


Figure 8: Percentage of the time duration of learned policy for phase Green-WE (green light on W-E and E-W direction, while red light on N-S and S-N direction) in every 2000 seconds for different methods under configuration 4.

5.6.3 Interpretation of learned signal. To understand what our method have learned w.r.t. dynamic traffic conditions, we show the percentage of duration for phase Green-WE (i.e., green light on *WE* direction with red light on *SN* direction), along with the ratio of traffic flow on *WE* over total traffic flow from all directions. With the changing of traffic, an ideal traffic light control method would be able to adjust its phase duration to traffic flows and get high reward. For example, as traffic changes from direction *WE* to *SN*, the traffic light agent is expected to adjust its phase duration from giving *WE* green light to giving *SN* green light. As we can see from Figure 8, *IntelliLight* can adjust its phase duration as the traffic changes.

5.7 Performance on Real-world Data

5.7.1 Comparison of different methods. In this section, we compare our method with baseline methods on real-world data. The overall results are shown in Table 10. Our method *IntelliLight* achieves the best reward, queue length, delay and duration over the best performance among baseline methods. Note that, the waiting time is reset to 0 once the car starts to move. Therefore, it is not directly related to other measures, like queue length. Due to the fact that we are optimizing a combination of different measures, it is reasonable that our method does not perform the best in terms of the waiting time. However, our method outperforms all other methods with regard to duration (the total time that one car spends before passing the intersection). This is one the most important measure that people care when they drive on the road.

5.7.2 Observations with respect to real traffic. In this section, we make observations on the policies we learned from the real data. We analyze the learned traffic light policy for the intersection of Jingliu Road (*WE* direction) and Erhuanxi Auxiliary Road (*SN* direction) under different scenarios: peak hours vs. non-peak hours, weekdays vs. weekends, and major arterial vs. minor arterial.

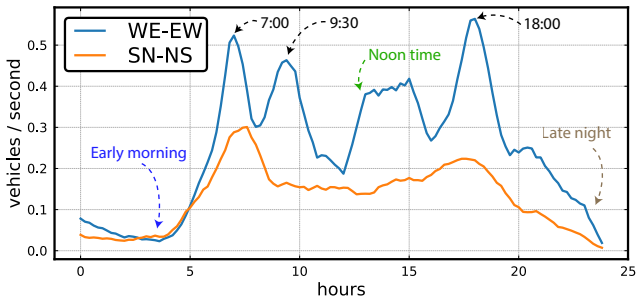
1. Peak hour vs. Non-peak hour. Figure 9 (a) shows the average traffic flow from both directions (*WE* and *SN*) on a Monday. On this day, there is more traffic on *WE* direction than *SN* for most of the time, during which an ideal traffic light control method is expected to give longer time for *WE* direction. It can be seen from Figure 9 (c) that, the ratio of the time duration for phase Green-WE (i.e., green light on *WE*, while red light on *SN*) is usually larger than 0.5, which means for most of the time, our method gives longer time for *WE*. And during peak hours (around 7:00, 9:30 and 18:00), the policies learned from our method also give longer time for green light on *WE* than non-peak hours. In early morning, the vehicle arrival rates on *SN* are larger than the rates on *WE*, and our method automatically gives longer time to *SN*. This shows our method can intelligently adjust to different traffic conditions.

2. Weekday vs. Weekend. Unlike weekdays, weekend shows different patterns about traffic condition and traffic light control policies. Our policy gives less green light on *WE* (more green light on *SN*) during weekend daytime than it gives on weekday. This is because there is more traffic on *SN* than on *WE* during weekend daytime in Figure 9 (b), while during weekday traffic on *SN* is less than on *WE*. Besides, by comparing Figure 9 (a) with Figure 9 (b), we can see that the traffic of *WE* and *SN* during late night time on Monday is similar, making the ratio of duration Green-We close to 0.5.

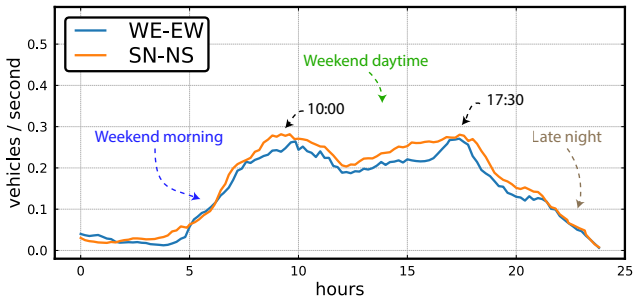
3. Major arterial vs. Minor arterial. Major arterials are roads that have higher traffic volume within a period, and expected to have a longer green light time. Without prior knowledge about major arterial, learned traffic light control policy using our method prefer giving the major arterial green light (including keeping the green light already on major arterial, and tend to switching red light to green light for major arterial). Specifically, we look into three periods of time (3:00, 12:00 and 23:30) of August 1st. From Figure 9 (a), we can tell that the road on *WE* direction is the main road, since traffic on *WE* is usually heavier than traffic on *SN*. As is shown in Figure 10, the dotted lines indicates the number of arriving cars for every second on two different directions. Along with the arrival

Table 10: Performances of different methods on real-world data. The number after ± means standard deviation. Reward: the higher the better. Other measures: the lower the better.

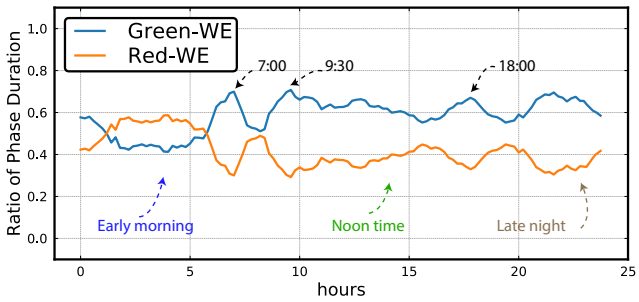
Methods	Reward	Queue Length	Delay	Waiting Time	Duration
FT	-5.727 ± 5.977	19.542 ± 22.405	3.377 ± 1.057	**3.969** ± 4.492	84.513 ± 60.888
SOTL	-35.338 ± 65.108	16.603 ± 17.718	4.070 ± 0.420	123.760 ± 244.463	64.833 ± 23.136
DRL	-30.577 ± 26.242	54.148 ± 43.420	4.209 ± 1.023	70.550 ± 67.644	166.861 ± 93.985
IntelliLight	**-3.892** ± 7.609	**10.238** ± 20.949	**2.730** ±1.086	5.169 ± 11.783	**50.487** ± 46.439



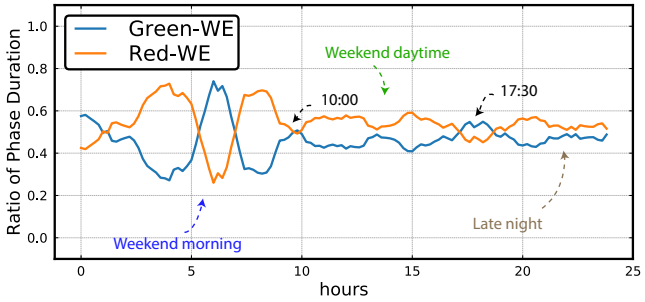
(a) Average arrival rate of August 1st (Monday)



(b) Average arrival rate of August 7th (Sunday)



(c) Phase time ratio from learned policy on August 1st (Monday)



(d) Phase time ratio from learned policy on August 7th (Sunday)

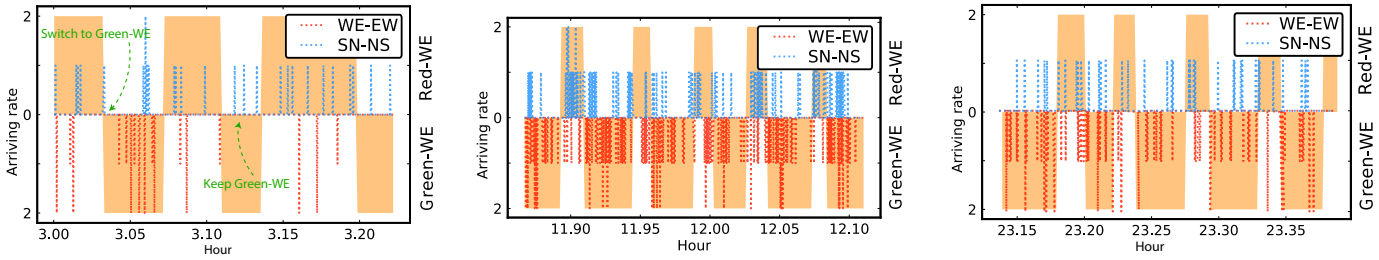
Figure 9: Average arrival rate on two directions (*WE* and *SN*) and time duration ratio of two phases (Green-WE and Red-WE) from learned policy for Jingliu Road (*WE*) and Erhuanxi Auxiliary Road (*SN*) in Jinan on August 1st and August 7th, 2016.

rate, we also plot the change of phases (dashed area). It can be seen from Figure 10 (a) that: 1) the overall time period of phase Red-WE is longer than Green-WE, which is compatible with traffic volume. 2) although the traffic volume of *SN* is larger than *WE*, the traffic light change from Green-WE to Red-WE is usually not triggered by waiting cars on *WE* direction. On the contrary, in Figure 10 (b) and Figure 10 (c), the change from Green-WE to Red-WE is usually triggered by waiting cars on *SN* direction. This is mainly because the road on *WE* is the main road during these time periods, and the traffic light tends to favor phase Green-WE.

6 CONCLUSION

In this paper, we address the traffic light control problem using a well-designed reinforcement learning approach. We conduct extensive experiments using both synthetic and real world experiments and demonstrate the superior performance of our proposed method over state-of-the-art methods. In addition, we show in-depth case studies and observations to understand how the agent adjust to the changing traffic, as a complement to quantitative measure on rewards. These in-depth case studies can help generate traffic rules for real world application.

We also acknowledge the limitations of our current approach and would like to point out several important future directions to make the method more applicable to real world. First, we can extend the two-phase traffic light to multi-phase traffic light, which will involve more complicated but more realistic state transition. Second, our paper addresses a simplified one intersection case, whereas the real world road network is much more complicated than this. Although some studies have tried to solve the multi-intersection problem by using multiple reinforcement learning agents, they do not explicitly consider the interactions between different intersections (i.e., how can the phase of one intersection affect the state of nearby intersections) and they are still limited to small number of intersections. Lastly, our approach is still tested on a simulation



(a) Early morning when traffic on *WE-EW* is less than *SN-NS*

(b) Noon when traffic on *WE-EW* is more than *SN-NS*

(c) Late night when traffic on *WE-EW* is more than *SN-NS*

Figure 10: Detailed average arrival rate on two directions (dotted lines) and changes of two phases (dashed areas) in three periods of time for Jingliu Road (*WE*) and Erhuanxi Auxiliary Road (*SN*) in Jinan on August 1st, 2016. X-axis of each figure indicates the time of a day; left Y-axis of each figure indicates the number of cars approaching the intersection every second; right Y-axis for each figure indicates the phase over time.

framework and thus the feedback is simulated. Ultimately, a field study should be conducted to learn the real-world feedback and to validate the proposed reinforcement learning approach.

ACKNOWLEDGMENTS

The work was supported in part by NSF awards #1544455, #1652525, #1618448, and #1639150. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

REFERENCES

[1] Monireh Abdoos, Nasser Mozayani, and Ana LC Bazzan. 2013. Holonic multi-agent system for traffic signals control. *Engineering Applications of Artificial Intelligence* 26, 5 (2013), 1575–1587.

[2] Baher Abdulhai, Rob Pringle, and Grigoris J Karakoulas. 2003. Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering* 129, 3 (2003), 278–285.

[3] Itamar Arel, Cong Liu, T Urbanik, and AG Kohls. 2010. Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems* 4, 2 (2010), 128–135.

[4] Bram Bakker, Shimon Whiteson, Leon Kester, and Frans CA Groen. 2010. Traffic light control by multiagent reinforcement learning systems. In *Interactive Collaborative Information Systems*. Springer, 475–510.

[5] Seung-Bae Cools, Carlos Gershenson, and Bart D'Hooghe. 2013. Self-organizing traffic lights: A realistic simulation. In *Advances in applied self-organizing systems*. Springer, 45–55.

[6] Francois Dion, Hesham Rakha, and Youn-Soo Kang. 2004. Comparison of delay estimates at under-saturated and over-saturated pre-timed signalized intersections. *Transportation Research Part B: Methodological* 38, 2 (2004), 99–122.

[7] The Economist. 2014. The cost of traffic jams. https://www.economist.com/blogs/economist-explains/2014/11/economist-explains-1.

[8] Samah El-Tantawy, Baher Abdulhai, and Hossam Abdelgawad. 2013. Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): methodology and large-scale application on downtown Toronto. *IEEE Transactions on Intelligent Transportation Systems* 14, 3 (2013), 1140–1150.

[9] Juntao Gao, Yulong Shen, Jia Liu, Minoru Ito, and Norio Shiratori. 2017. Adaptive Traffic Signal Control: Deep Reinforcement Learning Algorithm with Experience Replay and Target Network. *arXiv preprint arXiv:1705.02755* (2017).

[10] Wade Genders and Saiedeh Razavi. 2016. Using a deep reinforcement learning agent for traffic signal control. *arXiv preprint arXiv:1611.01142* (2016).

[11] Robert Godwin-Jones. 2010. Emerging technologies. (2010).

[12] Federico Guerrini. 2014. Traffic Congestion Costs Americans $124 Billion A Year, Report Says. *Forbes, October* (2014).

[13] Lior Kuyer, Shimon Whiteson, Bram Bakker, and Nikos Vlassis. 2008. Multiagent reinforcement learning for urban traffic control using coordination graphs. *Machine learning and knowledge discovery in databases* (2008), 656–671.

[14] Eric LG Legge, Christopher R Madan, Enoch T Ng, and Jeremy B Caplan. 2012. Building a memory palace in minutes: Equivalent memory performance using virtual versus conventional environments with the Method of Loci. *Acta psychologica* 141, 3 (2012), 380–390.

[15] Li Li, Yisheng Lv, and Fei-Yue Wang. 2016. Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica* 3, 3 (2016), 247–254.

[16] Mengqi Liu, Jiachuan Deng, Ming Xu, Xianbo Zhang, and Wei Wang. 2017. Cooperative Deep Reinforcement Learning for Tra ic Signal Control. (2017).

[17] Patrick Mannion, Jim Duggan, and Enda Howley. 2016. An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In *Autonomic Road Transport Support Systems*. Springer, 47–66.

[18] Alan J Miller. 1963. Settings for fixed-cycle traffic signals. *Journal of the Operational Research Society* 14, 4 (1963), 373–386.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[20] Isaac Porche and Stéphane Lafortune. 1999. Adaptive look-ahead optimization of traffic signals. *Journal of Intelligent Transportation System* 4, 3-4 (1999), 209–254.

[21] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.

[22] Elise van der Pol and Frans A Oliehoek. 2016. Coordinated Deep Reinforcement Learners for Traffic Light Control. NIPS.

[23] F. V Webster. 1958. Traffic signal settings. *Road Research Technical Paper* 39 (1958).

[24] MA Wiering. 2000. Multi-agent reinforcement learning for traffic light control. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*. 1151–1158.