Low-Cost Traffic Control Using Reinforcement Learning With Limited State

Ahmed F. AbouElhamayed, Hani Mahdi
Faculty of Engineering, Ain Shams University
Cairo, Egypt
ahmed.abouelhamayed@eng.asu.edu.eg, hani.mahdi@eng.asu.edu.eg
Cherif Salama
Faculty of Engineering, Ain Shams University
The American University in Cairo
Cairo, Egypt
cherif.salama@eng.asu.edu.eg

***Abstract*— Solving the traffic congestion problem has many benefits financially and environmentally. The application of Artificial Intelligence to solving the traffic congestion problem has been going on for a while. However, most of the current research in this area depends on knowing lots of information about all vehicles in the network. While it produces promising results, applying these techniques in the current world is not easy. In this paper, we apply reinforcement learning to the field of traffic control under the assumption that only minimal information is available. Our approach produces results that are better than currently deployed fixed-time traffic lights without having heavy requirements. In our first test configuration, our agent's waiting time is 82.3% of the best fixed-time traffic lights' waiting time and the average CO_2 emissions produced by our agent is 97.5% of the emissions produced by the best fixed-time traffic lights. This shows the potential of applying reinforcement learning to the Traffic Control problem with limited resources.**

Keywords—Artificial Intelligence; Reinforcement Learning; Traffic Control;

I. INTRODUCTION

Traffic congestion is one of the problems that have massive cost in our life. Some even call it a plague of modern life[1]. According to the World Economic Forum, traffic congestion costs the US economy alone nearly $87 billion in 2018[2].

Minimizing traffic congestion can be achieved by multiple ways ranging from very primitive but often costly solutions such as widening the roads to much more advanced techniques such as guiding all the vehicles in the network based on their sources and destinations to routes that would cause limited to no traffic congestion. Widening the road has its limitations as roads cannot be widened if there are existing buildings around the side. Guiding all vehicles on the other hand requires massive amount of information, processing, and connections.

The main contributions of this paper can be summarized as follows:

- Experimenting the efficiency of reinforcement learning in the traffic control problem given limited resources.
- Achieving a better average waiting time and carbon dioxide CO_2 emissions than fixed-time traffic lights in most tested configurations.
- Examining the effect of single vs multiple sensors per lane.

The remainder of this paper is organized as follows: Section II gives the necessary background on reinforcement learning which we used to train our agent. Section III describes the problem that we tackled in detail. The proposed model to solve the problem follows in Section IV. The experiment setup is explained in Section V followed by the results in Section VI. Finally, the paper is concluded in Section VII with some comments on the results and some suggested future work.

II. REINFORCEMENT LEARNING BACKGROUND

Machine learning can be classified into classical learning, ensemble methods, neural nets and reinforcement learning. Classical learning includes supervised and unsupervised learning. In this paper, we use reinforcement learning. In reinforcement learning, there is an agent that is given an environment and is asked to behave optimally in that environment. It learns by trying different actions and observing the rewards that result from these actions. Its goal is to try to maximize its rewards. In this section, we explore the details of reinforcement learning as it is the method used in this paper [3].

Consider an environment in which there is a specific goal that should be achieved. The environment has many states and it can be in any of those states. Consider $\boldsymbol{S}$ to be the set of all possible states in which the environment can be. An agent has the ability to take any action from a set of possible actions $\boldsymbol{A}$. taking an action $A_t \in \boldsymbol{A}$ while the environment is in a state $S_t \in \boldsymbol{S}$ moves the environment to be in a state $S_{t+1} \in \boldsymbol{S}$ and the agent takes a reward R_{t+1}. The agent's goal is to maximize its rewards over time giving preference to short-term rewards than long-term rewards. i.e. a reward at time $t+1$ is better than a reward at time $t+2$. We will be representing this as a value called the Return $\boldsymbol{R_t} = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$. where γ, ranging between 0 and 1, is known as the discount factor and it indicates how important are future rewards with respect to

present rewards. The higher its value, the more the agent cares about future rewards. If it is 0, then the agent only cares about the current reward and pays no attention to upcoming rewards. The agent should find a policy π which indicates for each state which action to take in order to maximize the return. An optimal policy $\pi^*$is a policy that gives you the maximum return possible from any state. Obtaining the optimal policy π^* is frequently a difficult task given that the agent only has the ability to try an action on the given environment and observe the reward from that action and the new state in which the environment is at as shown in Fig. 1.

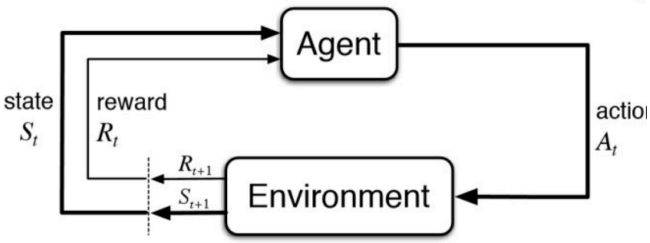

Fig. 1. Basic idea of reinforcement learning

To be able to find an approximation of the optimal policy π^*, let us define a new function $Q_\pi(S, A)$ which we will call the Quality of taking an action A when in state S under policy π. That value is the expected return if one takes action A and then follows policy π thereafter.

$$Q_\pi(S, A) = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = S, A_t = A\right]$$

Where E_π is the expected value of the equation if we followed policy π. We can define the optimal Q-function as the Q-function that has the maximum value from all possible policies as that gives the maximum return.

$$Q_*(S, A) = \max_\pi Q_\pi(S, A)$$

If you have the optimal Q-function, you can easily get the optimal policy from it; Given any state, you would search for the Q-value in that state's actions and choose to apply that action.

We can thus define the optimal Q function as follows:

$$Q_*(S, A) = E\left[R_{t+1} + \gamma \max_{A'} Q_*(S_{t+1}, A')\right]$$

Where E is the expected value of the equation and S_{t+1}, and R_{t+1} are the next state and the reward that are the results of applying action A in state S respectively. This equation is called the Bellman optimality equation. It states that for any state-action pair, the quality, or expected return, from doing that action on that state is equal to the reward returned and the discount factor multiplied by the maximum Q-value that can be obtained from the next state by applying all its actions.

Reinforcement learning is based on this equation and its goal is to build the optimal Q-function which is stored as a table in the memory. To build that table, we start by building an arbitrary table. We can for example start by setting all the values in the Q-function to 0. After that, we start applying random actions to the environment and as we apply actions, we start to learn some of the correct values of the Q-table and thus update the memory. After some iterations, the Q-table in the memory should converge to the optimal Q-function.

To decide on how long we should apply random actions and how frequent should our actions be random, we define a value ε called the exploration rate ranging between 0 and 1. It defines the probability of the agent taking random actions instead of using the existing Q-table to pick one. It is usually preferred to start with a large exploration value when the Q-table is arbitrary and gradually decrease this value as the Q-table is updated with more trustworthy values.

The Q-table is updated according to the following equation:

$$Q_{new}(S, A) = (1 - \alpha) Q_{old}(S, A) + \alpha\left(R_{t+1} + \gamma \max_{A'} Q(S_{t+1}, A')\right)$$

The value α, also ranging between 0 and 1, is called the learning rate and it defines how much we trust the new Q-value obtained and how big of a change we want to make in the existing Q-table based on it.

III. Problem Formulation

A. The Traffic Intersection

The problem of traffic control has many formats in the literature. Some papers study controlling a traffic light in a single simple intersection where the movement is either vertical or horizontal with no turns allowed like [4] and [5]. Others study more complex intersections where many possible movements between different lanes are allowed like [6], [7], [8] and [9]. Some papers study more than one intersection in the same problem such as [5] and [10]. This paper tackles the single simple intersection problem. Vehicles either move from the top lane to the bottom lane and vice versa or move from the left lane to the right lane and vice versa. The studied intersection is shown in Fig. 2 below.

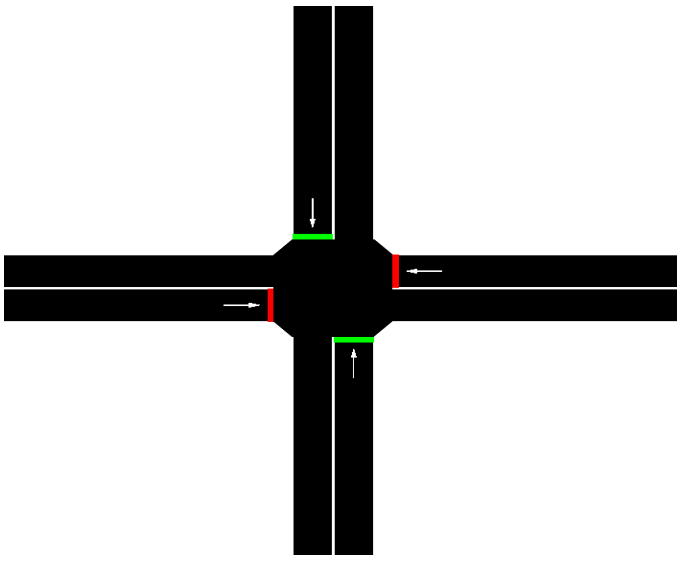

Fig. 2. The traffic intersection studied in this paper

B. Information Available to the Agent

Lots of the recent papers assume that information about all vehicles in the network is available. For example, the work in [5] and [11] assume that the position of all vehicles around the intersection is known and in [7] it is assumed that the positions and velocities of all vehicles around the intersection are known.

In this paper, we will consider that the agent has access to minimal amount of information. The goal is to use limited resources to allow the application of the technique practically without excessive cost. We will assume that there are sensors placed in some positions around the intersection and that these sensors along with the traffic light state itself are the only sources of information available to the agent. A similar approach can be seen in [10]. The values that will be extracted from the sensors are defined in Section IV.

C. Actions Doable by the Agent

In this paper, we will consider that the agent has the ability to control the traffic light only and has no control over the routes of the vehicles. Vehicles and their routes are generated without knowledge or control from the agent. The traffic light is said to have many phases. It can be in one of those phases. A phase is a combination of red/yellow/green lights in different directions. Some of the papers study the possibility of having variable times, set by the agent, for the yellow light like [12]. While others like [13] and [8] have a predefined time for yellow light. Some of the papers allow the agent to specify the phase of the traffic light at specific times like [14] but adds a limitation such that the action taken might not be applied immediately but a series of other phases might have to happen to allow that phase to keep the vehicles safe to have a yellow light in any direction while switching from green to red for example. In this model, the order of phases at different times is not guaranteed to be constant. In other papers like [6], the agent is only allowed to select when the next phase should happen but the ordering of phases is always preserved. In this paper, we have a fixed time for the yellow light and since there are actually 2 phases only that contain no yellow light, choosing the phase manually or choosing the go to the next phase is similar in our case but when going from green to red, yellow must happen first. The agent is not given control while the phase contains a yellow light and it is given control in all other times. Yellow time is fixed at 6 timesteps.

D. Goal

The goal of reducing traffic congestion is difficult to measure. [6] takes into account 3 criteria: number of arrived vehicles, average waiting time of a vehicle and the time loss for each vehicle. [5] measures the travel time and [10] measures the average number of vehicles waiting at stoplights during the simulation as well as the average amount of CO_2 emissions per distance travelled by the vehicles in the simulation.

In this paper, we consider the average waiting time per vehicle as well as the average CO_2 emissions per vehicle. Waiting time is calculated by adding up the number of vehicles at each timestep that have been waiting for more than one timestep. A vehicle is considered waiting if its speed is less than 0.1 meters/sec.

IV. Proposed Model

A. State

We have to derive the state from the sensors and the traffic light only to achieve the goal of minimizing the cost required to apply this research practically. Some papers like [5] use a binary matrix where 1 indicates that a vehicle is present in the position and 0 otherwise. While this representation is valid in their case as this matrix covers a wide area around the traffic intersection, it would not be sufficient in our case since we have access to a small number of positions, the ones that have sensors, and thus we need to include as much information as possible about these positions.

We are limited to information that can actually be calculated by the sensor. So, we cannot get the cumulative waiting time of the vehicle standing over the sensor since the sensor has no way of knowing the history of that vehicle. However, we can get the waiting time of the vehicle standing over it as the waiting time of a vehicle standing over a sensor can be easily calculated using software as long as the sensor can identify when a new vehicle is in its range. In addition to the information from the sensors, we add the current traffic light phase to the state to help the agent take the correct action.

One final element is added to the state, which is the duration of the current phase; The time that the traffic light has been in this phase. This is to help the agent know whether it should switch phases or not if both lanes have waiting vehicles.

If we keep the state as described above without any modifications, the state-space would be infinite as the numbers are continuous. To be able to apply Q-table without the need to go with Deep Q-Network to approximate it, we split the possible numbers into ranges and give a number to each range. The ranges and their mappings are shown in Table I. below.

TABLE I. Mapping of Ranges To Values In The State

Element	Mapping	
	Range	***Value***
Waiting Time of the vechicle on the sensor (in timesteps)	No Vehicles	0
	0 - 3	1
	3 - 20	2
	>= 20	3
Phase Duration (in timesteps)	0 – 3	0
	3 – 6	1
	6 – 10	2
	>= 10	3

B. Reward

The reward is one of the key components that affect the performance of the agent. Unlike the state, where we had to depend only on information from the sensors; In the calculation of the reward we can depend on any information available in the environment. The goal of the agent is to maximize the reward. However, in the traffic control problem, what we usually want to do is penalize the agent for congestion and so

the reward is most of the time a negative value or the inverse of another value. There are some exceptions when the reward is calculated as the throughput of the vehicles as the case in [15] or a function in the speed of the vehicles as in [16]. Some papers like [17] and [18] calculate the reward function as a function of queue lengths. Other papers like [19] consider multiple reward functions, some of them are functions of the delay experienced by the vehicles. Some papers consider multiple weighted factors in the reward such as [12] including penalties for switching the phase, waiting time, delay and emergency stops, which happen when the vehicle is asked to decelerate heavily, usually as a result of small yellow time. Some papers consider the reward as the difference between some function before applying the action and after it like [7].

In this paper, we use the negative value of the waiting time of all vehicles in all lanes as the reward function. Since our agent is only given choice when the traffic light is in a phase that contains a green light, i.e. a phase that does not contain a yellow light, when the action of the agent causes a phase with yellow light, the reward in that case is negative the sum of all the values of waiting times of all vehicles in all lanes in all the timesteps in which the phase contains a yellow light. This means that the reward when the agent causes a switch is much worse than when the agent keeps the phase as it is. This discourages the agent from causing a switch unless necessary without explicitly penalizing switching like [12].

C. Action Space

The agent is asked for an action in each time step in which the traffic light has no yellow light. This means that the agent does not control yellow times as mentioned earlier. The actions allowed are binary; 0 means keep everything as is and 1 means go to the next phase which necessarily means that the currently green lane is going to be red, after a yellow light phase, and the currently red lane is going to be green. Some of the papers like [10] give control to the agent only every fixed amount of timesteps but we are not following that and we are not putting any minimum time for the green light to let the agent learn it on its own and to not enforce the traffic light to have a green lane that has no vehicles while the other red lane has vehicles waiting for some time.

V. Experiments

Multiple experiments are carried out to test the performance of the proposed model using a simulated environment. Simulation of Urban Mobility (SUMO) [20] is used to carry out the experiments. It is a platform where the roads, routes, vehicles and traffic lights are defined. Traci [21], which provides an API to get information about vehicles and traffic lights as well as controlling the simulation and its components, is used to get information needed to provide the state, rewards, and performance measurements. It is also used to actuate the actions of the agent in the environment. Next is a description of each of the experiments that are carried out.

Some values are common in all experiments. Exploration rate ε is set to 1.0 initially and decreases by 0.01 per episode such that the agent starts by fully exploring and then moves to exploiting as it goes through new episodes and it reaches full exploitation after 100 episodes. The value of α is also set to 1.0 at the beginning and decreases by 0.005 per episode till it reaches 0.02 and stays at that value for the remainder of the training. The value of γ is 0.95. For the first 5000 timesteps, vehicles are generated according to specific probabilities defined for each of the experiments.

A. Experiment 1: One Sensor Per Lane

In this experiment, we have a total of 4 sensors and the size of the state space is $2^1 \times 4^4 \times 4 = 2048$. 2 is for the phase of the traffic light. Since it is limited to the phases where there is a green light. It has only 2 possible values. The 4^4 correspond to the possible values of the 4 sensors, while the last 4 corresponds to the duration of the current traffic light phase. The 4 sensors are each placed at a distance of 5 meters away from the end of the lane approaching the traffic light. It is the position at which the first vehicle waiting in the queue of that traffic light would stand.

For training, each episode takes one of configurations 1-3 shown in Table II in turns. The model is trained for 700 episodes. The average waiting time per vehicle for each episode is shown in Fig. 3. It is worth noting that at the beginning of the training the actions are mostly random due to high exploration rate and as episode 100, when exploration stops, is reached, the agent behaves much better than the random behavior at the beginning.

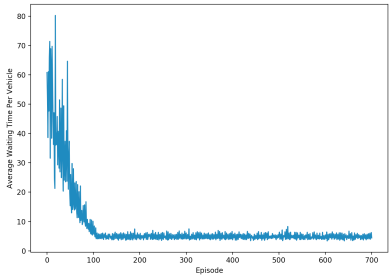

Fig. 3. Training performance for Experiment 1

B. Experiment 2: Two Sensors Per Lane

In this experiment, we have a total of 8 sensors and the size of the state space is $2^1 \times 4^8 \times 4 = 524{,}288$. The values are the same as the previous experiment with the addition of another 4^4 corresponding to each of the additional sensors. The 4 new sensors are each placed at a distance 50 meters away from the end of the lane approaching the traffic light to detect if the queue of waiting cars in each of the lanes has reached this length. The goal of this experiment is to assess the effect of increasing the number of sensors.

Training is done in the same way as the previous experiment. The average waiting time per vehicle for each episode is very similar to the one obtained from the previous experiment shown in Fig. 3.

Another set of experiments is performed for configurations 4, 5 and 6 which represent heavier traffic. For these experiments, a 2-sensor agent is trained sequentially for 500 episodes without changing the other parameters.

TABLE II. PROBABILITIES OF VEHICLES IN EACH CONFIGURATION

Configuration	Probabilities of vehicle generation	
	Route	***Probability***
Configuration 1	East to West / West to East	0.04
	North to South / South to North	
Configuration 2	East to West / West to East	0.04
	North to South / South to North	0.02
Configuration 3	East to West / West to East	0.02
	North to South / South to North	0.04
Configuration 4	East to West / West to East	0.1
	North to South / South to North	
Configuration 5	East to West / West to East	0.02
	North to South / South to North	0.1
Configuration 6	East to West / West to East	0.1
	North to South / South to North	0.02

VI. RESULTS

As recent researches are taking a different approach with higher cost data, our proposed solution is compared with the fixed-time traffic lights configuration to understand the efficiency of applying such a low-cost solution instead of the currently widespread fixed-time traffic lights. 3 different fixed-time traffic lights configurations are tested:

- Long Phase: 60 seconds between switching
- Medium Phase: 30 seconds between switching
- Small Phase: 15 seconds between switching

The yellow time for all of them and for the agent is fixed at 6 seconds. The simulation is run for 100 episodes on each of the models in comparison. During these episodes, no learning happens but the vehicles generation is different randomly. The results are reported in the Fig. 4. The x-axis contains the episodes and the y-axis contains the average waiting time per vehicle.

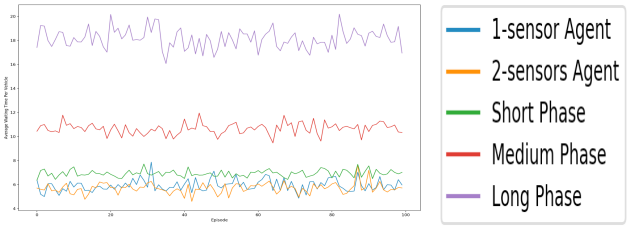

Fig. 4. Testing results for configuration 1.

The results show that both our agents outperform the fixed-time traffic lights. The performance is close to the Short Phase version of the fixed-time traffic lights and in one of the episodes, the fixed-time traffic lights beat the 1-sensor Agent. The results also show that 2-sensors Agent is better than 1-Sensor Agent 68% of the time.

The result of configuration 2 is shown in Fig. 5 and it is very similar to the result of configuration 3. Similar to the previous configuration, our agent outperforms the fixed-time traffic lights. In these configurations, the agents are always better than fixed-time traffic lights. Interestingly, the 1-sensor model outperformed the 2-sensors model in these configurations. The 2-sensors model performed better than the 1-sensor model only 28% and 41% of the times in configurations 2 and 3 respectively. In Fig. 6, it can be seen that the 2-sensor agent outperforms the Short Phase in CO_2 emissions as well. The x-axis contains the episodes and the y-axis contain the average CO_2 emissions per vehicle.

In Fig. 7, it can be seen that the Medium and Long Phases perform better than our agent for configuration 4. However, in configuration 5, our agent outperforms all others as shown in Fig. 8. A similar performance is observed in configuration 6.

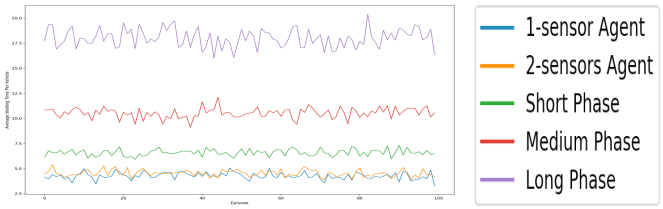

Fig. 5. Testing results for Configuration 2.

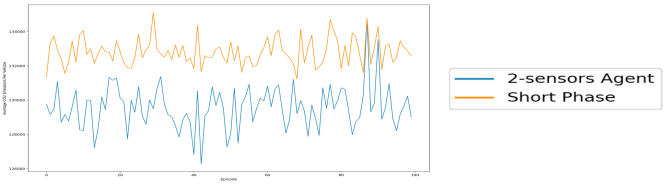

Fig. 6. CO_2 emissions for configuration 1.

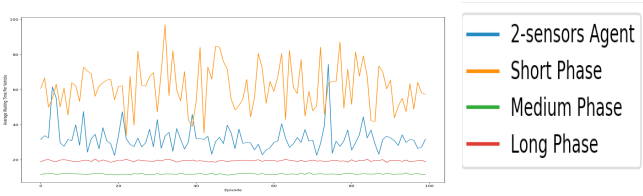

Fig. 7. Testing results for configuration 4.

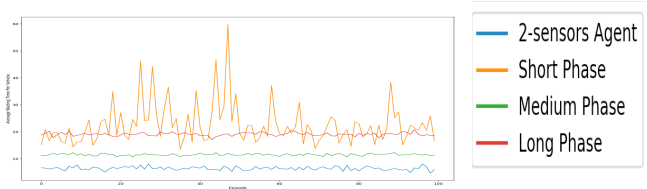

Fig. 8. Testing results for configuration 5.

VII. Conclusion And Future Work

In this paper, we showed the potential of applying reinforcement learning in the field of traffic control without needing much information and showed that it has high potential as our proposed model beat fixed-time traffic lights in the problems under test. We examined 2 different models with a different number of sensors. The models managed to score less average waiting time per vehicle for most of the configurations. We showed the effect of adding extra sensors which improved the performance in some cases and made it worse in some others given that both were given the same amount of training. This shows that adding more sensors would require more training so that it can reach the same performance of the less sensors model. The agent is outperformed by two of the fixed-phase traffic lights in case of heavy traffic. Observing the behavior of the agent shows that it does not perform optimally in case of traffic congestion on both approaching lanes as it switches quicker than the optimal. More training or tweaking of the reward function might be the solution to fixing this behavior. That trained model still beats all fixed-time traffic lights in all the other configurations.

Some of the methods used can be fine-tuned or replaced by other methods which might be a little more expensive or require more processing but might actually perform better. Fine-tuning the ranges used in the state-space and the positions of the sensors might lead to better performance. Replacing the Q-Table with a deep Q-Network can also improve the performance although it might require more training.

Acknowledgment *(Heading 5)*

The authors would like to acknowledge Instabug for hosting part of this research during September 2019 Hackweek.

We would also like to thank Ahmed Bassel, Mai Yehia, Mazen Magdy, and Youmna Elhassany for their contribution in this research.

References

[1. Arnott, R. and K. Small, *The economics of traffic congestion.* American scientist, 1994. **82**(5): p. 446-455.

2. Forum, W.E. *Traffic congestion cost the US economy nearly $87 billion in 2018*. 2019; Available from: https://www.weforum.org/agenda/2019/03/traffic-congestion-cost-the-us-economy-nearly-87-billion-in-2018/.

3. Sutton, R.S. and A.G. Barto, *Introduction to reinforcement learning*. Vol. 2. 1998: MIT press Cambridge.

4. Mousavi, S.S., M. Schukat, and E. Howley, *Traffic light control using deep policy-gradient and value-function-based reinforcement learning.* IET Intelligent Transport Systems, 2017. **11**(7): p. 417-423.

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

6. Lin, Y., et al., *An efficient deep reinforcement learning model for urban traffic control.* arXiv preprint arXiv:1808.01876, 2018.

7. Liang, X., et al., *Deep reinforcement learning for traffic light control in vehicular networks.* arXiv preprint arXiv:1803.11115, 2018.

8. Genders, W. and S. Razavi, *Using a deep reinforcement learning agent for traffic signal control.* arXiv preprint arXiv:1611.01142, 2016.

9. Gao, J., et al., *Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network.* arXiv preprint arXiv:1705.02755, 2017.

10. Stevens, M. and C. Yeh, *Reinforcement learning for traffic optimization*. 2016, Stanford. edu.

11. Rijken, T., *DeepLight: Deep reinforcement learning for signalised traffic control*. 2015, Master's Thesis. University College London.

12. van der Pol, E., *Deep reinforcement learning for coordination in traffic light control.* Master's thesis, University of Amsterdam, 2016.

13. Nishi, T., et al. *Traffic Signal Control Based on Reinforcement Learning with Graph Convolutional Neural Nets*. in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018. IEEE.

14. El-Tantawy, S. and B. Abdulhai. *An agent-based learning towards decentralized and coordinated traffic signal control*. in *13th International IEEE Conference on Intelligent Transportation Systems*. 2010. IEEE.

15. Salkham, A.a., et al. *A collaborative reinforcement learning approach to urban traffic control optimization*. in *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Volume 02*. 2008. IEEE Computer Society.

16. Casas, N., *Deep deterministic policy gradient for urban traffic light control.* arXiv preprint arXiv:1703.09035, 2017.

17. Abdoos, M., N. Mozayani, and A.L. Bazzan. *Traffic light control in non-stationary environments based on multi agent Q-learning*. in *2011 14th International IEEE conference on intelligent transportation systems (ITSC)*. 2011. IEEE.

18. Zheng, G., et al., *Diagnosing Reinforcement Learning for Traffic Signal Control.* arXiv preprint arXiv:1905.04716, 2019.

19. El-Tantawy, S., B. Abdulhai, and H. Abdelgawad, *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, 2013. **14**(3): p. 1140-1150.

20. Krajzewicz, D., et al., *Recent development and applications of SUMO-Simulation of Urban MObility.* International Journal On Advances in Systems and Measurements, 2012. **5**(3&4).

21. Wegener, A., et al. *TraCI: an interface for coupling road traffic and network simulators*. in *Proceedings of the 11th communications and networking simulation symposium*. 2008. ACM.