From Local to Global: A Curriculum Learning Approach for Reinforcement Learning-based Traffic Signal Control

Nianzhao Zheng
Waseda University
Tokyo, Japan
protoss@akane.waseda.jp

Jialong Li
Waseda University
Tokyo, Japan
lijialong@fuji.waseda.jp

Zhenyu Mao
Waseda University
Tokyo, Japan
rockmao@akane.waseda.jp

Kenji Tei
Waseda University
Tokyo, Japan
ktei@aoni.waseda.jp

2022 IEEE 2nd International Conference on Software Engineering and Artificial Intelligence (SEAI) | 978-1-6654-8223-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/SEAI55746.2022.9832372

***Abstract*—Traffic signal control (TSC) is one of the effective ways to mitigate traffic congestion, a growing problem causing significant economic loss to urban areas. In recent years, there is an increasing interest in using reinforcement learning (RL) in TSC since RL shows great potential in optimizing control policy for complex real-world traffic conditions. However, one concerning problem is the huge computing time/resources required to train the control policy for large-scale TSC. To address this problem, this paper proposes a method that utilizes a curriculum to help speed up the training process. Control policies of different single-intersection maps are learned first and then referenced to a large-scale map consisting of a certain number of different intersections. The preliminary evaluation demonstrates that our method achieves a jump-start compared to that of learning the target task from scratch.**

***Index Terms*—traffic signal control, curriculum learning, reinforcement learning**

I. Introduction

Traffic congestion has become a growing problem, causing much inconvenience to people's daily life [1]. TSC is one of the most studied and expected ways to mitigate traffic congestion and improve traffic conditions. Recently, there is a growing interest to apply reinforcement learning (RL) techniques to TSC systems [2]. An RL TSC system does not rely on the rule of control designed by a human. By interacting with the complex environment, RL agents could learn an efficient control policy and achieve better performance than humans can do.

However, applying such techniques usually lead to a common problem in machine learning. It often requires very high computation time/resources for trial-and-error attempts. A large number of update iterations of RL models is needed to get satisfactory results. Especially, the above problem is critical for a large-scale TSC system. As the scale of the TSC system becomes larger, with more vehicles, lanes and intersections, the state variables in the system increase. This could cause the state explosion problem, which means the size of the system state space would grow exponentially. Thus, the trial-and-error attempts required during the training process would also increase exponentially, leading to an increase in computing resources. Such attempts may cause real traffic jams in the TSC problem. [3].

In this paper, we propose a method that utilizes a curriculum for reinforcement learning-based traffic signal control (RL-based TSC). By creating a curriculum that orders the tasks from simple to difficult and performing knowledge transfer between them, it is possible to achieve faster convergence and even improve the performance on a difficult task. [4] Usually, lots of intersections in a large-scale TSC system could have high similarities. The localized knowledge learned from training a single traffic signal junction could be easily reused for multiple intersections during the training for a multi-intersection TSC system. This could help globally reduce the trial-and-error attempts required for the whole TSC system and time reduction can be expected through such method.

The contributions of this paper are concluded as follows:

- We apply a curriculum learning (CL) method to an RL-based TSC problem.
- We propose a design methodology for generating intermediate tasks according to the characteristics of TSC.

The rest of the paper is organized as follows. Section 2 gives the background. Section 3 discusses the existing works. The proposal is made in Section 4 and then the experiment results are shown in Section 5. Finally, we conclude the paper and discuss the future work in Section 6.

II. Background

A. Reinforcement Learning-based Traffic Signal Control

For an RL-based TSC problem, it could be stated in the form of a Markov Decision Process (MDP) $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$. $\mathcal{S}$ represents the state space, a set of all possible states the agent can transition to. For the domain of traffic signal control, elements such as queue length [5], waiting time of vehicles [6] or current traffic light phase [7] are widely used as the quantitative description of the environment state $\mathcal{A}$ is action space, which is a set of all actions the agent can take in a specific environment. The action usually could be choosing the next phase [8] or setting the duration of current phase [9]. $\mathcal{P}$ is the probabilistic state transition function, $\mathcal{R}$

978-1-6654-8223-3/22/$31.00 ©2022 IEEE

is the reward function and $\gamma(0 \leq \gamma \leq 1)$ is the discount factor. At time step t, a reinforcement learning agent in state $s_t \in \mathcal{S}$, takes an action $a_t \in \mathcal{A}$, progresses into the next state $s_{t+1} \in \mathcal{S}$ and obtains a reward r_t by a reward function of $r_t = \mathcal{R}(s_t, s_{t+1}, a_t)$. Based on the traffic information acquired, traffic signal control aims to adjust the duration of green phases and avoid congestion by shortening the waiting time of the vehicles or ensuring the capacity of the road is not exceeded. [3]

The goal of an RL agent is to learn policy π that maximizes the expected return. In our work, Q-learning, a standard RL algorithm, is used. The policy π is learned by performing value iteration update defined as follows:

$$Q^{new}(s_t, a_t) = Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

where α is the learning rate $(0 < \alpha \leq 1)$. Through this way, the agent may find an optimal policy π^* that maximizes the Q-function. [10]

B. Curriculum Learning for Reinforcement Learning

Recent RL research has been exploring how to utilize transfer learning, reusing the knowledge learned from a source task to a target task. A sequence of tasks produces a curriculum as knowledge is transferred from one task to another. Narvekar et al. concluded the method of CL with the point of view that "In the most general case, where the agent can acquire experience from multiple intermediate tasks that differ from the final MDP, there are 3 key elements to this method". [4] First is **task generation**. To build a curriculum we need to create the parts that compose the curriculum first, which is a set of intermediate tasks. The aim is to exploit the training on the final target task by knowledge transfer through those intermediate tasks. The second is **sequencing**, which means to create a partial ordering over the set of experience samples. Most existing works would define curricula manually, which means a human selects the order of tasks or samples, while recent works also consider automatic sequencing methods. The third is **transfer learning**. Since the MDP setting of intermediate tasks usually differs from the final task such as the state space and action space, we need to decide what knowledge is to be reused and how it can be reused. Knowledge could be Q-function, a policy or a task model. The agent would repeatedly perform knowledge transfer from one task to the next until a set of final tasks.

III. Related Work

Reinforcement learning-based approaches: Many reinforcement learning-based control approaches have been widely used in the field of TSC. At first, foundational RL algorithms, for example, Q-learning [11] and Sarsa [12] are applied. As more elements are introduced into state definitions, deep RL algorithms such as Deep Q Network [13] [14], a typical value-based deep RL algorithm, are utilized to handle complex environments and demonstrate their ability to solve complicated traffic signal control problems. Rizzo et al. [15] worked on a policy-based deep RL that a utilize deep neural network to learn a policy that maps from the input state to a distribution of actions. To adapt to the heterogeneous scenarios in TSC faster, Zang et al. [16] proposed a value-based meta-reinforcement learning method and improve the efficiency of training RL agents in TSC. Huang et al. [17] combine model-based reinforcement learning algorithm and meta-learning method for TSC to further reduce the iterations required for training. However, to our knowledge, none of the existing works have considered exploiting curriculum in the domain of TSC. Our work proposed a method combining curriculum learning (CL) and reinforcement learning-based traffic signal control. For the domain of TSC, we also specially design a method for the task generation of the curriculum.

Multi-agent reinforcement learning: Some studies also addressed Multi-agent RL (MARL) in the multi-intersection TSC problem [8]. Usually, each agent controls a single traffic signal junction and all of the agents are trained from scratch. In our work, we utilize knowledge reuse to reduce the trial-and-error attempts required for the whole system, improving the learning efficiency of the multi-intersection TSC problems.

IV. Curriculum Learning for RL-based TSC Problem

A. Overview

Utilizing curriculum learning could help the agents learn a difficult task more effectively when some training is done on a simple task and the knowledge is transferred from the simple task to a difficult task. In our situation, the difficulty of the task depends on the size of the TSC system: the larger the size, the more complex the state-action combination. We will utilize an existing Q-learning-based TSC. Each agent controls one traffic light. Taking information such as the status of traffic light and numbers of vehicles queuing on the incoming lanes as the observation, an agent chooses a green phase for a traffic light as the next action, trying to maximize the reward (calculated from delay in relation to the previous step). Then we will apply our CL method to the Q-learning-based TSC to learn an optimized control policy for multi-intersection situations. With knowledge transfer, reusable knowledge learned from a single intersection environment is extracted and passed on to the next task, up to the final target TSC system. Former works have defined the curriculum as a directed acyclic graph of intermediate tasks [18] [19], which is suitable for our situation. The curriculum diagram for our method is shown in Fig.1. Except for the common intersection types such as crossroad, T-junction or Y-junction, different number of lanes also form different types of intersection, such as type 1 and type 2 in Fig.1.

By utilizing such method, a jump-start could be expected since agents do not have to learn from scratch but continue the learning based on the policy learned from previous tasks. Many works also have shown that curriculum may even help improve the performance on the final task, as well as reduce the time needs to converge to an optimal policy.

In the rest part of this section, details of three key elements of curriculum learning for reinforcement learning: task gen-

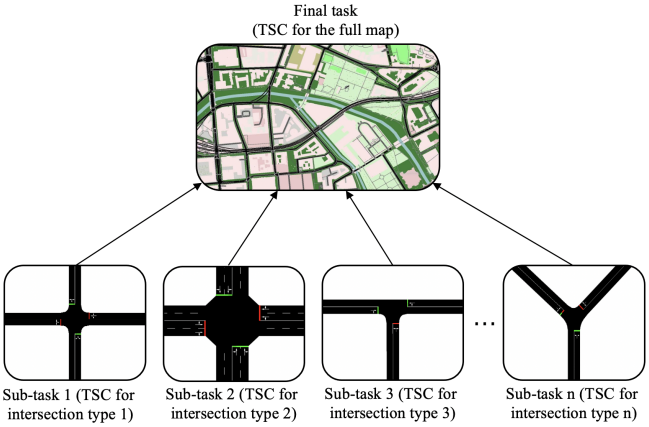

Fig. 1: Curriculum diagram

eration, sequencing and knowledge transfer are described to explain how they are achieved in our proposal.

B. Task Generation

One basic idea to generate intermediate tasks is creating easier solvable versions of the final task, which can be achieved by reducing the size of the TSC system in our case. Similar or even identical intersections usually exist in a large-scale TSC system. Every single intersection is a small part of the whole TSC system. We can consider learning the control policy of a single intersection as an intermediate task, which is a simple version of learning the control policy of the whole system. By dividing the map of the final target TSC system into small parts up to a single intersection and then identifying the different types of intersections or a combination of them, each different type would be considered as the environment of an intermediate task.

C. Sequencing

In our work, we choose to manually define the curriculum. The sequencing of our situation could be considered simply as ordering in complexity. This means each intermediate task's environment is a part of its next task's environment. For example, one intermediate task is learning a control policy on one intersection, then the next task to which the learned knowledge will be passed should be learning a control policy on a TSC system including that type of intersection, or directly the final target TSC system.

D. Transfer Learning

What kind of knowledge is reusable from one intermediate task to the next is one of the most important things we need to decide when creating a curriculum. Since our proposal is a Q-learning-based TSC and the observation and action space of each similar intersection remain the same, the most intuitive and effective knowledge for our case would be Q-function. After finishing the training of single agents, information of the status of the neighbor traffic lights is added to create communication between agents for the multi-intersection situation, enabling them to behave as a group in a multi-agent system. Although the actions remain the same, the state would be different between sub-tasks and the final task. However, the state of an intersection under the multi-intersection situation can be considered as an extension of the state of an intersection under the single-intersection situation. Two parts consist of the state of an intersection under a multi-intersection situation: the local state part and the communication part. The local state part is the state space of a single intersection environment and the communication part will be the neighbor traffic lights' status, specifically the current active green phase. Utilizing this characteristic, we could perform knowledge transfer between sub-tasks and the final task.

Algorithm 1 shows the detail of how Q-function is transferred. The input is the Q-function learned from a sub-task and the output will be the initial Q-function for the same type intersection in the final task. We first give the definition of state for the local state part, which is the state for a single intersection task:

Definition 1 (Local state)**.** The local state s for single intersection is defined as a 3-tuple $< P, Q, D >$ where $P = \{north\_south,\ east\_west,\ etc.\}$ represents the current active green phase, Q represents the number of queued vehicles in incoming lanes, D is the number of queued in the vehicles in incoming lanes divided by the maximum capacity of the lanes.

Then we define the global state for the final tasks:

Definition 2 (Global state)**.** The global state $< s, s_n >$ of an intersection for the final task is a 2-tuple where s is the local state and s_n is an n-tuple $< P_1, P_2, ..., P_n >$ that represents the current active green phase all of the neighbor traffic lights.

For all local state s learned from a sub-task (line 1), the Q-function value $Q(< s, s_n >, a)$ will be initialized with the same value as the learned Q-function value $Q(s, a)$ that has the same local state to it (lines 2-6). In this way, the Q-function $Q(< s, s_n >, a)$ is reusable for the final task, adapting to the new task's state and action.

Take a regular crossroad that has four same crossroad neighbour traffic lights as an example, its s_n would be a four-tuple $< P_1, P_2, P_3, P_4 >$ denotes the current phase of the neighbor traffic lights in four directions. For such a crossroad that has two possible green phases: north-south green and east-west green, we could represent the green phases with 0 and 1. In addition, -1 is used if there is no neighbor traffic light in that direction. This leads up to 3^4 possible combinations of the status of neighbor traffic lights, which means 3^4 same Q-function value with different communication part but same local state part and action would be created. For example, assuming we have $s_1 = s'_1$, and since actions should be the same, we represents the actions all with a. Then for an input learned from a sub-task with state-action combination $(s, a) = (s_1, a)$, the initial Q-function for the final task with state-action combination$(< s, s_n >, a) = (< s'_1, s_n >, a)$ is transferred with the same value. Here $s_n = s_{n_1}, s_{n_2}, ...s_{n_i} (i = 3^4 = 81)$ correspond to all the possible combinations of the status of

neighbor traffic lights.

Algorithm 1 Q-function transfer

Input: Q-function learned from sub-task $Q(s, a)$
Output: Initial Q-function $Q(< s, s_n >, a)$ for final task

```
for all local state s do
    for all possible combinations of status of neighbor traffic lights s_n do
        Q(< s, s_n >, a) ← Q(s, a)
    end for
end for
return Q(< s, s_n >, a)
```

V. Preliminary Evaluation

A. Experiment Setting

We set our experiment to examine the effectiveness of our proposal. Our research question is set as:

- **RQ1**: Can the CL method achieve a jump-start compared to learning from scratch and how much could our method improve the initial performance of the agents on the final task?
- **RQ2**: What kind of scenarios can our method have a good effect and what kind of scenarios our method is not good at?

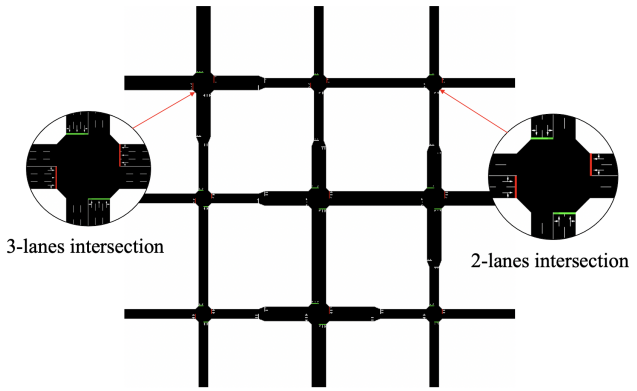

Fig. 2: 3x3 grid network with example of two different types of intersections shown inside circle

As shown in Fig.2, our final target task is a 3x3 grid network that has two different types of intersections in it. Both of them are crossroads with the same number of lanes in four directions. The first type of intersection consists of four incoming approaches and four outgoing approaches with two lanes in each. The second type also has four incoming approaches and four outgoing approaches but has three lanes in each approach. We first train agents on the networks consisted only one single intersection of each type as our intermediate tasks, then the knowledge is transferred to learn the final task.

Considering the size of our map, we keep 100 vehicles running in the simulation for our final task. 3600 seconds of simulation time in the simulator is considered one training episode. Considering that there could be safety concerns if the switch of traffic light phases is too frequently, to guarantee the safety, t = 5s is set so that the environment is simulated for 5 seconds after each MDP step. This means the agent can only choose its next action, specifically the next green phase configuration, after 5 seconds of its last action, resulting in an episode horizon of 720 steps. ϵ-greedy is used as the behavior policy for all tasks. For intermediate tasks, the ϵ is set to 0.5 and is adjusted to 0.1 for the final task. The learning rate α is set to 0.1 and the discount factor γ is 0.9.

To evaluate the effectiveness of our proposal, we compare the performance between the CL method and the RL method. The average travel time and delay time of the vehicles finished their trip are used to compare the performance. Travel time is the time a vehicle spent to finish its trip and delay time is the stopped time a vehicle spent waiting for a green light. Such information is obtained based on a simple interface for reinforcement learning environments with SUMO simulator built by Lucas [20]. The average delay time and travel time of vehicles finishing their trips in each episode will be calculated. We then examine whether there is a jump-start between two different methods. We will also see if the final performance is improved through the CL method.

B. RQ1

We trained 20 times for each method and show the results in Fig.3 and Fig.4. Both figures' horizontal axis represents the training episode and the vertical axis of Fig.3 and Fig.4 represent the average travel time and average delay time of the vehicles of each episode respectively. Two lines show the average of all training results while the shaded area represents the standard deviation. For the RL method, the initial random policy cause an average travel time of about 162 seconds and an average delay time of about 47 seconds as the blue line shows. For the CL method, the transferred Q-function helped improve the initial performance to an average travel time of about 143 seconds and an average delay time of about 30 seconds as the orange line shows. It is about a 29-seconds improvement in average travel time and a 17-seconds improvement in average delay time at the beginning of the training, which indicates achieving an obvious jump-start compared to learning from scratch. The result floats within a certain range after some episodes' training, which should be caused by the different route lengths for each vehicle. Both methods showed a convergence of around 130 seconds and 18 seconds for average travel time and average delay time, but the CL method did not have an apparent improvement in the performance of our final task. In conclusion, experiment results show that our CL method can achieve a jump-start by significantly improving the agents' initial performance on the final task. A considerable reduction is realized for the travel time and delay time of vehicles at the beginning of the training.

C. RQ2

For the current implementation, it is only available for highly similar intersections, which is not relatively reliable

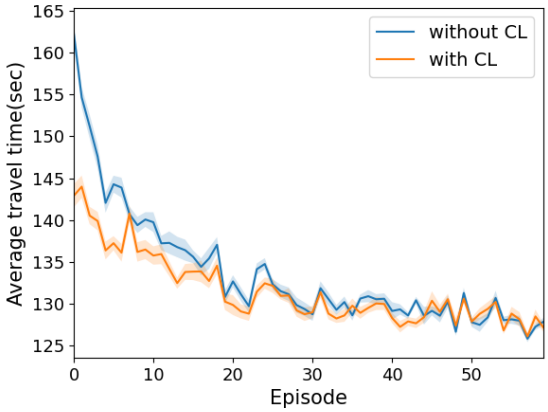

Fig. 3: Average travel time in 3x3 grid network

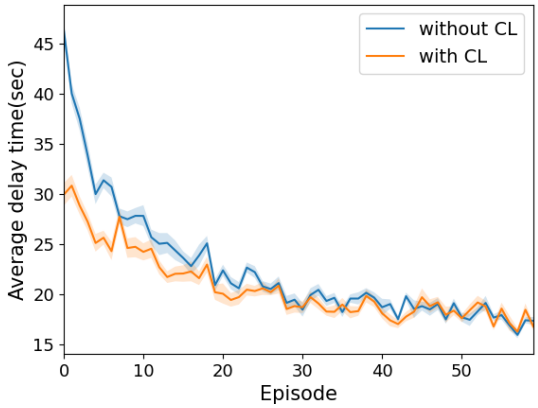

Fig. 4: Average delay time in 3x3 grid network

considering the possible complex real-world conditions. While applying such method to a typical grid network consisting of limited types of crossroads, our method should have great effectiveness. Only a few intermediate tasks are required but the knowledge can be reused several times inside the grid network. The knowledge learned from intermediate tasks could also be used for different final tasks. When we need to train agents on a new TSC system, knowledge learned from previous learning may still be utilized. However, there are also real-world traffic scenarios that have more complex situations. Not only the number of lanes could have various situations, for example, the number of lanes in the north-south direction is different from that in the east-west direction, different types such as T-junction or Y-junction would also exist in a complicated TSC system. Under such scenarios, our method may not have high efficiency to reduce the trial-and-error attempts need for the whole system.

D. Discussion

Although a jump-start is achieved, our preliminary result did not show an obvious improvement in the performance of the final task as some other CL research suggested. There is also not much difference in training episodes required for convergence between the two methods. The complexity of the final task we set up is not high, which may be the reason caused the shortage described above. If we want to examine the capability of the CL method to improve the performance in a complicated task or shorten the training time to a certain threshold, further experiments with more complex curriculum or complicated final task may be required. For example, testing our method on a larger TSC system including more road junction types or simulating with heavier traffic flow.

VI. Conclusion and Future Work

This paper proposed a CL method for training a TSC system on a large-scale map to speed up the training process. We have specifically designed the curriculum for the domain of TSC and set an experiment with a final task on a 3x3 grid network to examine the effectiveness of our method. The experiment results show that it achieves an obvious jump-start, which means it has a good effect on initial performance improvement. We believe that our CL method could be expected to improve the learning efficiency for large-scale traffic signal control.

For future work, we plan to make an extension to our current work so that it could better adapt to the real-world situation. These include 1) testing our CL method with more powerful RL algorithms such as Deep Q network; 2) testing our CL method with more complex situations: larger, more complicated maps or heavier traffic flow; 3) proposing a transfer method for different intersections so that the knowledge transfer could be available for not just highly similar intersections.

Acknowledgment

The research was partially supported by JSPS KAKENHI and JSPS Research Fellowships for Young Scientists. The authors would like to thank Mr. Kun Liu for his comments on the early stage of this study.

References

[1] "Traffic snarled near greater tokyo at end of 3-day weekend," https://www.asahi.com/ajw/articles/14444707, 2021.

[2] P. Mannion *et al.*, "An experimental review of reinforcement learning algorithms for adaptive traffic signal control," in *Autonomic Road Transport Support Systems*, 2016.

[3] H. Wei *et al.*, "A survey on traffic signal control methods," *ArXiv*, vol. abs/1904.08117, 2019.

[4] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *J. Mach. Learn. Res.*, vol. 21, pp. 181:1–181:50, 2020.

[5] M. Abdoos, 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)*. IEEE, 2011, pp. 1580–1585.

[6] 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 International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2496–2505.

[7] S. El-Tantawy, 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*, vol. 14, no. 3, pp. 1140–1150, 2013.

[8] T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 1086–1095, 2019.
[9] M. Aslani, M. S. Mesgari, and M. Wiering, "Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events," *Transportation Research Part C: Emerging Technologies*, vol. 85, pp. 732–752, 2017.
[10] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html
[11] B. Abdulhai, R. Pringle, and G. J. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," *Journal of Transportation Engineering*, vol. 129, no. 3, pp. 278–285, 2003.
[12] T. L. Thorpe and C. W. Anderson, "Tra c light control using sarsa with three state representations," Citeseer, Tech. Rep., 1996.
[13] T. Rijken, "Deeplight: Deep reinforcement learning for signalised traffic control," Ph.D. dissertation, Master's thesis. University College London, 2015.
[14] E. Van der Pol 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.
[15] S. G. Rizzo, G. Vantini, and S. Chawla, "Time critic policy gradient methods for traffic signal control in complex and congested scenarios," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1654–1664.
[16] X. Zang, H. Yao, G. Zheng, N. Xu, K. Xu, and Z. Li, "Metalight: Value-based meta-reinforcement learning for traffic signal control," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 1153–1160.
[17] X. Huang, D. Wu, M. Jenkin, and B. Boulet, "Modellight: Model-based meta-reinforcement learning for traffic signal control," *arXiv preprint arXiv:2111.08067*, 2021.
[18] M. Svetlik *et al.*, "Automatic curriculum graph generation for reinforcement learning agents," in *AAAI*, 2017.
[19] P. MacAlpine *et al.*, "Overlapping layered learning," *Artif. Intell.*, vol. 254, pp. 21–43, 2018.
[20] L. N. Alegre, "SUMO-RL," https://github.com/LucasAlegre/sumo-rl, 2019.