Hierarchically and Cooperatively Learning Traffic Signal Control

Bingyu Xu[1], **Yaowei Wang**[1], **Zhaozhi Wang**[2], **Huizhu Jia**[2], **Zongqing Lu**[2†]
[1]Peng Cheng Laboratory
[2]Peking University

Abstract

Deep reinforcement learning (RL) has been applied to traffic signal control recently and demonstrated superior performance to conventional control methods. However, there are still several challenges we have to address before fully applying deep RL to traffic signal control. Firstly, the objective of traffic signal control is to optimize average travel time, which is a delayed reward in a long time horizon in the context of RL. However, existing work simplifies the optimization by using queue length, waiting time, delay, etc., as immediate reward and presumes these short-term targets are always aligned with the objective. Nevertheless, these targets may deviate from the objective in different road networks with various traffic patterns. Secondly, it remains unsolved how to cooperatively control traffic signals to directly optimize average travel time. To address these challenges, we propose a hierarchical and cooperative reinforcement learning method–HiLight. HiLight enables each agent to learn a high-level policy that optimizes the objective locally by selecting among the sub-policies that respectively optimize short-term targets. Moreover, the high-level policy additionally considers the objective in the neighborhood with adaptive weighting to encourage agents to cooperate on the objective in the road network. Empirically, we demonstrate that HiLight outperforms state-of-the-art RL methods for traffic signal control in real road networks with real traffic.

Introduction

Traffic signals coordinating traffic movements are the key for transportation efficiency. However, conventional traffic signal control that heavily relies on pre-defined rules and assumptions on traffic conditions is far from intelligence. Recently, researchers have started to apply deep reinforcement learning (RL) to traffic signal control and some studies (Wei et al. 2018; Zheng et al. 2019; Wei et al. 2019b,a) have obtained superior performance to conventional control methods (Mirchandani and Head 2001; Roess, Prassas, and McShane 2004; Varaiya 2013; Cools, Gershenson, and D'Hooghe 2013). The main advantage of RL is directly interacting with highly dynamic environment to learn to reach a long-term goal for traffic signal control.

However, there are still several challenges we have to address before we can successfully apply deep RL to traffic signal control. One major challenge is the discrepancy between the target of RL method and the objective of traffic signal control. Basically, the objective of traffic signal control is to optimize average travel time of all vehicles in a road network, which is a delayed reward in a long time horizon in the context of RL. However, existing work simplifies the problem and tailors its own target by using queue length, waiting time, delay, etc., as the reward. For example, IntelliLight (Wei et al. 2018) uses a combination of these as the reward. Nevertheless, optimizing the cumulative rewards of such is not equivalent to minimizing average travel time. These targets can be aligned with the objective, but they can also substantially deviate from it in different road networks with various traffic patterns, which leads to the sub-optimality of existing RL methods.

Another major challenge is how to cooperatively control traffic signals to optimize average travel time. Many existing RL methods independently control traffic signals, such as IntelliLight (Wei et al. 2018) and MetaLight (Zang et al. 2020). Some cooperatively control traffic signals, such as CoLight (Wei et al. 2019b), PressLight (Wei et al. 2019a), and MPLight (Chen et al. 2020). At each intersection, CoLight uses additional information from neighboring intersections to optimize the queue length, while PressLight and MPLight implicitly consider the cooperation between neighboring intersections by minimizing the pressure which is loosely defined as the number of vehicles in incoming lanes minus the number of vehicles on outgoing lanes at an intersection. However, how to cooperatively and directly optimize average travel time in a road network is unclear.

To address these challenges, we propose a hierarchical and cooperative reinforcement learning method, HiLight, for traffic signal control. To ease the difficulty of directly optimizing average travel time, HiLight enables each agent to learn a high-level policy that minimizes average travel time locally by selecting between the learned sub-policies that respectively optimize different short-term targets, such as queue length, waiting time, and delay. By such a hierarchy, the high-level policy can easily learn to align the target of each sub-policy with its own objective while avoiding the difficulty of directly optimizing average travel time. To enable agents to cooperate on optimizing average travel time

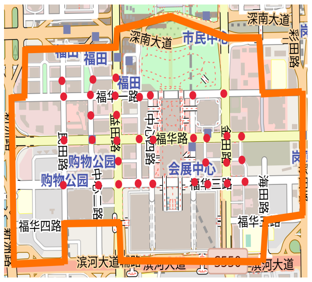
†Correspondence to Zongqing Lu <zongqing.lu@pku.edu.cn>
Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

in a road network, the high-level policy additionally optimizes average travel time in the neighborhood. Multi-critic is introduced to unify these two objectives with adaptive weighting and thus learn the cooperative policy in a natural way. We conduct extensive experiments in four real road networks with real traffic and empirically demonstrate HiLight significantly outperforms conventional control methods and state-of-the-art RL methods. By ablation studies, we verify the effectiveness of the hierarchy, multi-critic, and adaptive weighting.

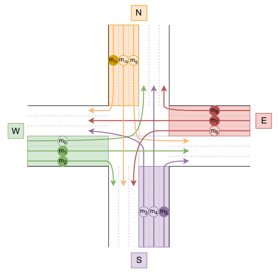
Related Work

Conventional traffic signal control depends on handcrafted rules, *e.g.*, fixing time interval between two phases in steady traffic condition (Roess, Prassas, and McShane 2004), pre-defining rules to trigger traffic signal by real-time traffic (Mirchandani and Head 2001; Cools, Gershenson, and D'Hooghe 2013), or setting a fixed offset among all intersections along an arterial for multi-intersection coordination (Roess, Prassas, and McShane 2004). However, these methods heavily rely on human knowledge, as they require manually designed traffic signal plans or rules. More recently, max pressure control (Varaiya 2013) is proposed for multi-intersection coordination, which minimizes the pressure of an intersection. However, it still relies on assumptions to simplify traffic condition.

RL based traffic signal control learns to control traffic signal by interacting with the environment, which does not require any pre-defined plans or rules and has shown superior performance to conventional control methods in many recent studies (Wei et al. 2018; Zheng et al. 2019; Wei et al. 2019a,b; Chen et al. 2020). Some studies consider each intersection isolated in a road network, *i.e.*, each intersection is controlled by an agent independently. These methods differ in terms of environment settings including state (Wei et al. 2018; Xu et al. 2019), action (change to next phase (Wei et al. 2018) or set a phase (Zheng et al. 2019; Chen et al. 2020)) and reward (queue length (Zheng et al. 2019), delay (El-Tantawy and Abdulhai 2012), or/and waiting time (Wei et al. 2018)), learning algorithms (Aslani, Mesgari, and Wiering 2017; Liang et al. 2018), and policy adaptation (Xu et al. 2019; Zang et al. 2020). These methods can easily scale up to multi-intersection scenarios. However, as they ignore the interaction between neighboring intersections in a road network, they indeed cannot optimize their objectives in the road network. To jointly optimize the objective in the road network, one can use centralized optimization to coordinate intersections (Van der Pol and Oliehoek 2016) or use centralized learning decentralized execution paradigms, such as QMIX (Rashid et al. 2018) and QTRAN (Son et al. 2019). However, as the road network scales up, centralized learning is infeasible due to exponentially increased joint action space. Therefore, other studies aim to enable agents to learn cooperation in a decentralized way by acquiring information from neighboring intersections (El-Tantawy and Abdulhai 2012; Nishi et al. 2018; Wei et al. 2019b) or optimizing a chosen reward that induces cooperation between agents, *i.e.*, pressure (Wei et al. 2019a; Chen et al. 2020). However, these



(a) Multi-intersection road network



(b) Movement and phase setting

Figure 1: Illustration of multi-intersection road network and movement and phase setting. The intersection in Figure 1b has 12 movements: m_0-m_{11} (turn left/right, go straight). A phase consists of several non-conflicting movements; *e.g.*, phase 1 is composed of m_1, m_2, m_5, m_7, m_8, and m_{11}, which means vehicles can go straight W-E and turn right.

methods still do not work well, mainly because their optimization objectives, such as queue length and pressure, are not always aligned with average travel time of all vehicles in the road network.

Hierarchical RL can solve complex tasks with sparse rewards or long time horizons. It learns multiple levels of policies, of which the higher level gives goals or options to the lower level and the lowest level applies actions to the environment. Therefore, the higher levels are able to plan over a long time horizon or a complex task. Existing studies consider learning sub-goals (Vezhnevets et al. 2017; Nachum et al. 2018), options (Bacon, Harb, and Precup 2017; Frans et al. 2018), or optimizing multiple objectives (Jiang and Lu 2019). However, none of these hierarchical RL methods can be directly applied to learning cooperative traffic signal control.

Preliminary

In this paper, we investigate traffic signal control in the scenario of multi-intersection as illustrated in Figure 1a. To illustrate the definitions clearly, we use the intersection with four approaches as an example as depicted in Figure 1b. However, the definitions can be easily extended to different kinds of intersections.

Intersection and Road Network Each intersection is controlled by traffic signals as shown in Figure 1b. The intersection consists of four incoming approaches and four outgoing approaches, namely East, South, West, North incoming approaches and East, South, West, North outgoing approaches. Each incoming approach consists of three lanes. From inner to outer, three lanes of each approach represent the "left-turn", "straight", "right-turn" directions.

Movements and Phase Generally, the vehicles of each incoming approach are going to turn right/left, or go straight. Considering that a vehicle can turn right at any time, we define 12 movements of an intersection, m_0 to m_{11}, as shown in Figure 1b. As some movements may conflict (*e.g.*, m_1

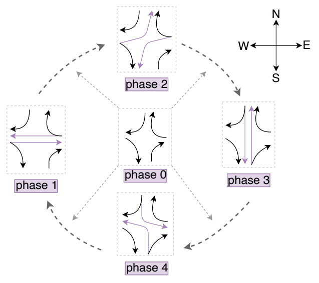


Figure 2: The main loop consists of phases 1 to 4, when changing phase from one to another, phase 0 (yellow signal) is added.

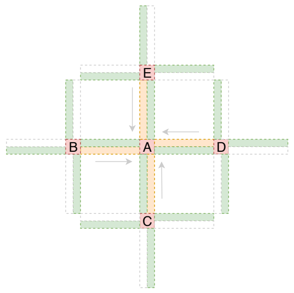


Figure 3: Local area of an intersection is yellow colored, and neighborhood area is both yellow and green colored.

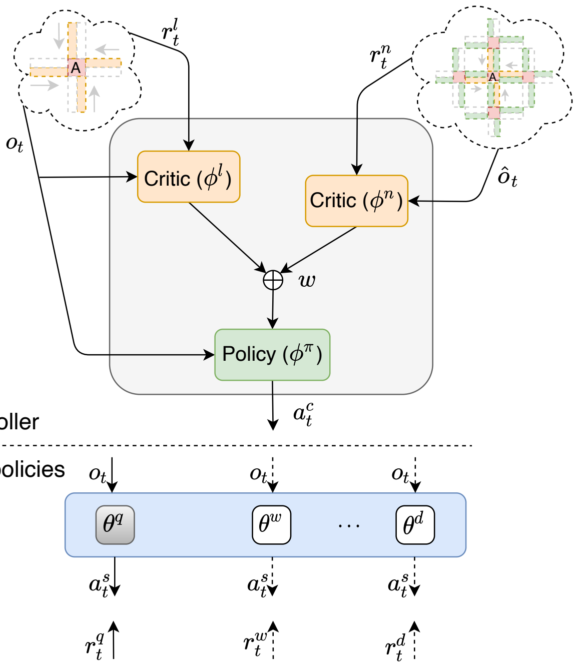


Figure 4: HiLight architecture. Each intersection is controlled by an agent which has one controller and several sub-policies. Every T timesteps, the controller selects a sub-policy to interact with the environment. The selected sub-policy gets a reward every timestep, while the controller gets a reward every T timesteps. The policy of the controller is learned based on two critics, which are weighted adaptively for optimizing local and neighborhood travel time.

from west to east and m_3 from south to west), only non-conflicting movements can be combined into one phase. We define five phases for a four-approach intersection (phase setting of different kinds of intersections may be different): phase 0 includes m_2, m_5, m_8, and m_{11}; phase 1 includes m_1, m_2, m_5, m_7, m_8, and m_{11}; phase 2 includes m_0, m_2, m_5, m_6, m_8, and m_{11}; phase 3 includes m_2, m_4, m_5, m_8, m_{10}, and m_{11}; phase 4 includes m_2, m_3, m_5, m_8, m_9, and m_{11}. The phase cycle is illustrated in Figure 2, where the main loop consists of phases 1 to 4 and phase 0 (yellow signal) is added when changing phase from one to another.

Average Travel Time The travel time of a vehicle is the time discrepancy between entering and leaving a particular area. Average travel time of all vehicles in a road network is the most frequently used measure to evaluate the performance of traffic signal control. However, as it can only be measured over a long time horizon, it is hard to optimize average travel time directly. In this paper, each agent (intersection) considers two other versions of average travel time: local travel time and neighborhood travel time. Local travel time of an intersection is average time vehicles spent on the local area of the intersection, while neighborhood travel time is average time vehicles spent on the neighborhood area of the intersection, which are both illustrated in Figure 3.

Method

The basic idea of HiLight is to align different targets that are easy to optimize using RL with the objective of traffic signal control in road networks. However, since average travel time of all vehicles in a road network is hard to optimize directly, HiLight instead optimizes local travel time. Further, to encourage cooperation towards optimizing average travel time, each agent additionally optimizes neighborhood travel time. HiLight is a hierarchy with a multi-critic controller, where two objectives are adaptively weighted to learn a policy for cooperative traffic signal control.

Hierarchy

Each intersection is controlled by a HiLight agent that has a hierarchical architecture as illustrated in Figure 4. The hierarchy consists of a controller and several sub-policies. Every T timesteps, the controller selects one of the sub-policies and the chosen sub-policy directly interacts with the environment for the next T timesteps. The controller gets a reward every T timesteps while the sub-policy get a reward every timestep. Therefore, the controller can focus on the long-term objective while the sub-policies concentrate on short-term targets. The hierarchy well suits traffic signal control. Average travel time is a statistic over a long time horizon, and thus can be optimized by the controller. The sub-policies focus on the targets that can be immediately and easily measured, such as queue length and waiting time, and be aligned with the long-term objective. In the following, we describe the design of the sub-policies and controller in detail.

Specialized Sub-Policies The sub-policies can optimize any targets that can be aligned with the optimization of average travel time. In current instantiation of HiLight, three sub-policies are employed to respectively optimize queue length, waiting time, and delay. However, they are not limited to these targets and can include any additional targets, *e.g.*, pressure (Wei et al. 2019a). Assuming there are $\mathcal{N}$ intersections in a road network, at each timestep, the sub-policy of each agent gets an observation o, which is the concatenation of three vectors: current phase, next phase, and the number of vehicles of the incoming lanes. Current and next phases are one-hot vectors, and the next phase is determined by the phase cycle. The action of the sub-policy a^s is to stay in the current phase or change to the next phase for next timestep. Note that the action is not limited to whether to change to the next phase, but can also be selecting next phase as in other studies (Zheng et al. 2019; Wei et al. 2019b; Chen et al. 2020). The rewards for the three sub-policies are the negatives of the sum of queue length of all incoming lanes (r^q), the sum of waiting time of all vehicles on the incoming lanes (r^w), and the sum of delay of all incoming lanes (r^d), respectively. The queue length is the total number of waiting vehicles on an incoming lane. The waiting time is the total time a vehicle in the waiting position (the speed of vehicle is below 0.1m/s). The delay of a lane is the difference between maximum speed and lane speed, divided by maximum speed. The three sub-policies are parameterized by θ^q, θ^w, and θ^d, respectively, and learned using DQN (Mnih et al. 2015) by minimizing the loss (the superscripts are dropped for generality),

$$\mathcal{L}(\theta) = \mathbb{E}[(r + \gamma \max_{a'} Q(o', a'; \theta) - Q(o, a; \theta))^2], \quad (1)$$

where o' denotes the next observation. Note that the sub-policies can also be learned using other RL methods.

Multi-Critic Controller The action of the controller, denoted as a^c, is to choose which sub-policy to interact with the environment for next T timesteps. The reward of the controller is the negative of local travel time, r^l. However, our objective is to optimize average travel time of all vehicles in the road network. Individually optimizing local travel time cannot guarantee that. It is because if each controller only considers its local travel time, their policies may conflict in terms of minimizing average travel time. Therefore, the controller is designed to additionally optimize an additional reward, neighborhood travel time r^n. From Figure 3, we can see that neighborhood travel time of intersection A is a statistic of local travel time of all the intersections in its neighborhood including itself. That said, optimizing neighborhood travel time takes neighboring agents' policies into account, towards better performance in a larger area. As the neighborhood areas of adjacent intersections overlaps, agents tend to cooperate on jointly optimizing their neighborhood travel time, which eventually leads to all agents cooperating on minimizing average travel time in the road network.

To enable the controller to jointly optimize both local and neighborhood travel time, we adopt actor-critic RL method and introduce multi-critic. Specifically, the controller has two value networks $V^l(o; \phi^l)$ and $V^n(\hat{o}; \phi^n)$ that respectively approximate the value function of the policy $\pi(a^c|o; \phi^\pi)$ in terms of local travel time and neighborhood travel time. The value functions and the policy are parameterized by ϕ^l, ϕ^n, and ϕ^π, respectively. Note that the input of V^n is different from V^l and π, where $\hat{o}$ is the concatenation of the observations of the intersections in the neighborhood. For example, in Figure 3, $\hat{o}$ at A is the concatenation of all the observations of A, B, C, D, and E. As V^n needs to learn the value function in terms of neighborhood travel time, V^n can approximate it more accurately by taking $\hat{o}$ as input instead of o. With the two critics, the policy gradient

Algorithm 1 HiLight training

```
 1: Initialize controller $\boldsymbol{\phi}$, sub-policies $\boldsymbol{\theta}$, and $w$ for each agent
 2: for episode = $1, \ldots, \mathcal{M}$ do
 3:     for agent =$1, \ldots, \mathcal{N}$ do
 4:         The controller chooses one sub-policy $\theta$
 5:         for $t = 1, \ldots,$ max-episode-length do
 6:             The chosen sub-policy $\theta$ acts to the environment
                and gets the reward $\begin{cases} r_t^q & \text{if } \theta = \theta^p, \\ r_t^w & \text{if } \theta = \theta^w, \\ r_t^d & \text{if } \theta = \theta^d, \end{cases}$
 7:             Update $\theta$ using update rule (1)
 8:             if $t\%T = 0$ then
 9:                 The controller gets rewards $r_t^l$ and $r_t^n$
10:                 Update $\phi^l$ and $\phi^n$
11:                 Update $\phi^\pi$ with weight $w$ using (2)
12:                 Update $w$ using (3)
13:                 The controller re-selects one sub-policy
14:             end if
15:         end for
16:     end for
17: end for
```

of $\pi(a^c|o; \phi^\pi)$ is

$$\nabla \phi^\pi = \mathbb{E}\left[\log \pi(a^c|o; \phi^\pi)(\delta^l + w\delta^n)\right], \quad (2)$$

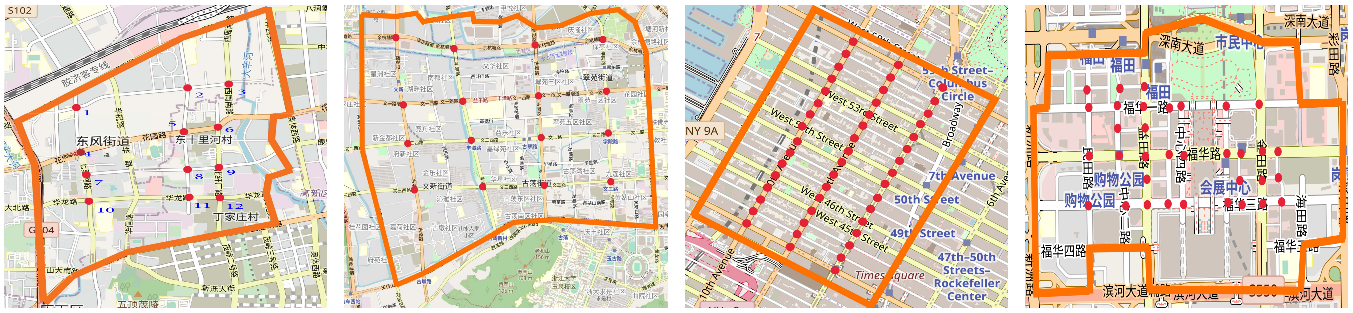
where $\delta^l = r^l + \gamma V^l(o'; \phi^l) - V^l(o; \phi^l)$ and $\delta^n = r^n + \gamma V^n(\hat{o}'; \phi^n) - V^n(\hat{o}; \phi^n)$ are the advantages of the two value functions respectively, and w is a weighting parameter.

Adaptive Weighting

The weight w needs to be tuned to balance these two objectives. However, manually tuning w incurs several issues. First, the importance of optimizing neighborhood travel time might change under different traffic patterns, and thus using a fixed value w may limit the performance. Second, if the value of w is fixed, then it can be learned by hyperparameter optimization. However, the learning process has to be run to near-convergence many times to determine the optimal value of w. This dramatically increases the computation.

To address these problems, we adapt the adaptive weighting mechanism (Lin et al. 2019) to enable the controller to learn the weight online to dynamically balance these two objectives during the learning process. Let $\mathcal{L}(\phi_i^\pi) = \mathcal{L}^l(\phi_i^\pi) + w\mathcal{L}^n(\phi_i^\pi)$, where $\mathcal{L}^l(\phi_i^\pi) = \delta^l$, $\mathcal{L}^n(\phi_i^\pi) = \delta^n$, and ϕ_i^π are the model parameters of the policy at training iteration i. As policy gradient is gradient ascent, we have $\phi_{i+1}^\pi = \phi_i^\pi + \alpha \nabla_{\phi_i^\pi} \mathcal{L}(\phi_i^\pi)$, where α is the learning rate of ϕ^π. We aim to find the weight w where $\mathcal{L}^l$ decreases the fastest. Specifically, let us define $s_i(w)$ as the speed at which $\mathcal{L}^l$ decreases at iteration i, and then we have,

$$\begin{aligned} s_i(w) = \frac{\mathrm{d}\,\mathcal{L}^l(\phi_i^\pi)}{\mathrm{d}\,i} &\approx \mathcal{L}^l\left(\phi_{i+1}^\pi\right) - \mathcal{L}^l\left(\phi_i^\pi\right) \\ &= \mathcal{L}^l\left(\phi_i^\pi + \alpha \nabla_{\phi_i^\pi} \mathcal{L}\left(\phi_i^\pi\right)\right) - \mathcal{L}^l\left(\phi_i^\pi\right) \\ &\approx \mathcal{L}^l\left(\phi_i^\pi\right) + \alpha \nabla_{\phi_i^\pi} \mathcal{L}^l\left(\phi_i^\pi\right)^{\mathrm{T}} \nabla_{\phi_i^\pi} \mathcal{L}\left(\phi_i^\pi\right) - \mathcal{L}^l\left(\phi_i^\pi\right) \\ &= \alpha \nabla_{\phi_i^\pi} \mathcal{L}^l\left(\phi_i^\pi\right)^{\mathrm{T}} \nabla_{\phi_i^\pi} \mathcal{L}\left(\phi_i^\pi\right), \end{aligned}$$



(a) Dongfeng sub-district, Jinan, China (b) Gudang sub-district, Hangzhou, China (c) Manhattan, New York City, USA (d) Fuhua sub-district, Shenzhen, China

Figure 5: Experimental road networks

where the third line is the first-order Taylor approximation. Then, we can calculate the gradient of w as,

$$\nabla_w s_i(w) = \alpha \nabla_{\phi_i^\pi} \mathcal{L}^l \left(\phi_i^\pi\right)^{\mathrm{T}} \nabla_{\phi_i^\pi} \mathcal{L}^n \left(\phi_i^\pi\right). \tag{3}$$

Thus, w is simply updated by the dot product of the gradients of $\mathcal{L}^l$ and $\mathcal{L}^n$ by gradient ascent. With adaptive weighting, HiLight learns to dynamically adjust w during learning.

Decentralized Training

During training, the critic for neighborhood travel time of each agent needs the observations of neighbors and neighborhood travel time. However, such information can be easily obtained, and thus HiLight can be easily learned in a decentralized way. Moreover, since the controller is learned using actor-critic, such information is not needed during execution. The training process of HiLight is detailed in Algorithm 1. HiLight is learned end-to-end, where the subpolices are learned based on DQN (Mnih et al. 2015) and the controller is learned based on PPO (Schulman et al. 2017).

Hyperparameter	HiLight	PressLight	CoLight
discount (γ)	0.9	0.8	0.8
batch size	128	20	20
buffer capacity	2048	1×10^4	1×10^4
sample size	512	1000	1000
ϵ and decay	0.4/0.97	0.8/0.95	0.8/0.95
T	50	–	
optimizer	Adam	RMSprop	RMSprop
learning rate	0.0001	0.001	0.001
learning rate of V^l	0.001	–	
learning rate of V^n	0.001	–	
# convolutional layers	–		3
# MLP layers	3	4	2
# MLP units		$(32, 32)$	
# MLP layers of V^l and V^n	3	–	
# MLP units V^l and V^n	$(32, 32)$	–	
MLP activation		ReLU	
initializer		random normal	

Table 1: Hyperparameters

Experiments

We conduct the experiments on CityFlow (Zhang et al. 2019), an open-source simulator for large-scale traffic signal control. The simulator runs on the traffic data, provides the state for the traffic signal control method, executes its action, and returns the feedback. HiLight is investigated based on four real datasets (real traffic data and road networks) and compared with both conventional and state-of-the-art RL methods for traffic signal control.

Settings

Datasets The four real datasets are from four cities including Manhattan, New York City in US and Jinan, Hangzhou, and Shenzhen in China. The road networks are illustrated in Figure 5. The road networks and traffic flows of Jinan, Hangzhou, and New York City are the public datasets[1].

The road networks of Jinan and Hangzhou contain 12 and 16 intersections in 4×3 and 4×4 grids, respectively. The traffic flow is generated from surveillance camera data. The road network of Manhattan includes 48 intersections in a 16×3 grid. The traffic flow is generated from open-source taxi trip data. The traffic of the three road networks has a time span of 3600 seconds. The road network of Shenzhen contains 33 intersections, imported directly from OpenStreeMap. All the intersections controlled by traffic signals in the enclosed area in Figure 5d are included. The traffic flow is generated from surveillance camera data. There are two sets of traffic data: one weekday and one weekend day, both of which have a time span of 10000 seconds. Moreover, the road network of Shenzhen is not grids, different from other three road networks.

Baselines We compare HiLight with both conventional and RL methods[2]. For ablation studies, we also compare with the variants of HiLight to verify the effectiveness of each component of HiLight. For fair comparison, the action of all RL methods is to decide whether to change to next phase in the phase cycle, and the action interval is five sec-

[1]https://traffic-signal-control.github.io/

[2]Some existing RL methods for traffic signal control, *e.g.*, MPLight (Chen et al. 2020), cannot run successfully on the latest CityFlow, and some targets on single intersection, *e.g.*, MetaLight (Zang et al. 2020). Therefore, they are omitted for comparison.

Method	**Average Travel Time** (seconds)					**Throughput**				
	Jinan	Hangzhou	Manhattan	Shenzhen (D)	Shenzhen (E)	Jinan	Hangzhou	Manhattan	Shenzhen (D)	Shenzhen (E)
FixedTime	749	762	1649	3821	3886	3399	1947	192	32	187
SOTL	1350	1413	1686	3854	3972	1480	2119	148	20	147
MaxPressure	397	352	443	3838	3873	5465	10428	2349	26	198
PressLight	382	425	1369	2220	1444	5517	9971	587	841	2246
CoLight	320	359	278	1574	1099	5688	10394	2615	1267	2999
IQLQ	381	386	292	531	665	5516	10251	2568	2126	3581
IQLW	437	470	866	817	758	5424	9673	1381	2197	3555
IQLD	452	736	526	797	795	5059	7279	1907	2140	3082
LocalCritic	346	343	271	395	440	5665	10529	2637	2370	3849
NBHDCritic	485	516	412	574	453	5137	9217	2228	2200	3826
StaticWeight	313	260	262	379	285	5725	11011	2631	2377	3968
HiLight	**290**	**252**	**251**	**337**	**223**	**5769**	**11063**	**2635**	**2482**	**4088**

Table 2: Performance of all methods in Jinan, Hangzhou, Manhattan, and Shenzhen with weekday (D) and weekend (E) traffic in terms of average travel time and throughout. From top to down are conventional methods, RL based methods, ablations of HiLight, and HiLight. The performance of all the RL methods is the average of three training runs with different random seeds after convergence.

onds for each method. Moreover, we use parameter-sharing for each kind of intersections in the road network to ease the training for all the RL methods.

• FixedTime (Koonce and Rodegerdts 2008) uses a predefined plan for cycle length and phase time, which is widely used for steady traffic.

• SOTL (Cools, Gershenson, and D'Hooghe 2013) specifies the upper and lower limit time of each phase. When the lower limit is reached, it will continue to detect whether there is a vehicle arriving. If there are no more vehicles arriving, the phase will be changed; otherwise, the green light time will be prolonged until no more vehicles arrive or the phase duration reaches the upper limit.

• MaxPressure (Varaiya 2013) selects the phase that maximizes the pressure.

• PressLight (Wei et al. 2019a) is an RL method that optimizes the pressure of each intersection.

• CoLight (Wei et al. 2019b) is an RL method that exploits graph convolutional reinforcement learning (Jiang et al. 2020) to use information from neighbors to optimize the queue length.

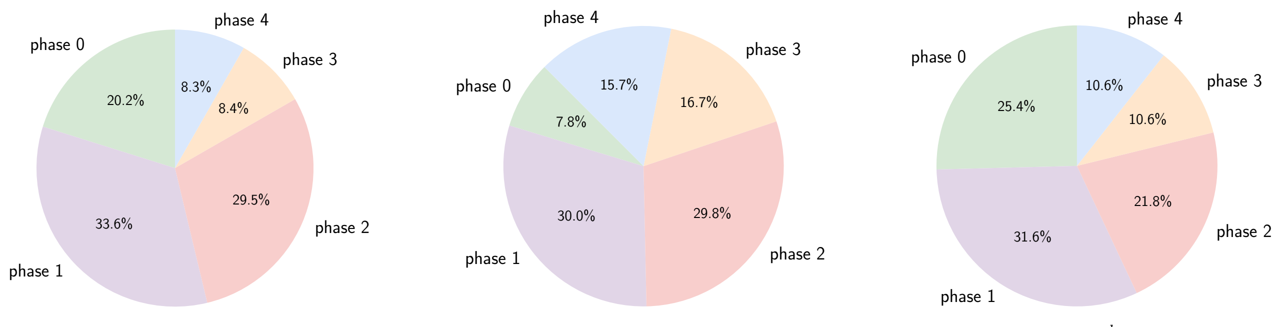
• IQL optimizes queue length, waiting time, or delay at each intersection using DQN, respectively denoted as IQLQ, IQLW, and IQLD, which are the sub-policies of HiLight and used for ablation studies.

• LocalCritic is HiLight with only the critic for local travel time, which is used for ablation studies.

• NBHDCritic is HiLight with only the critic for neighborhood travel time, which is used for ablation studies.

• StaticWeight is HiLight with a static w for each intersection, which is used for ablation studies.

Hyperparameters Table 1 summarizes the hyperparameters of HiLight, CoLight, and PressLight. For CoLight and PressLight, we use their open-source implementations. For fair comparison, we also use their default parameter settings which perform better than other settings as verified by experiments.



(a) θ^q (b) θ^w (c) θ^d

Figure 6: Behaviors of the sub-policies of an intersection in the road network of Jinan, in terms of percentage of time taken by each phase in an episode. The sub-policies indeed develop different policies to control traffic signals.

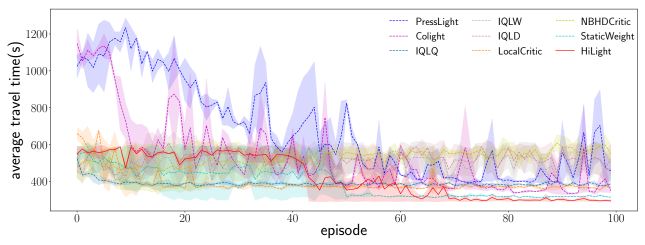
Evaluation Metrics Following the previous studies, we use *average travel time* of all vehicles and *throughput* of the road network to evaluate the performance of different methods for traffic signal control. The travel time of a vehicle is the time span between entering and leaving the road network. The throughput is the number of vehicles which have finished their routes over the course of the simulation.

Performance Comparison

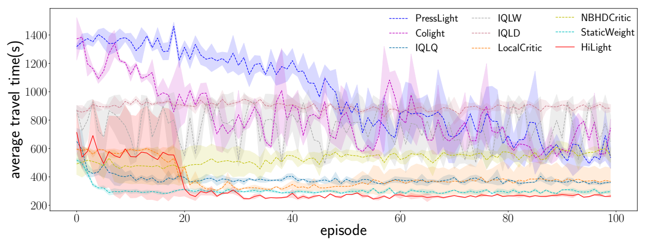
The performance of all the baselines in four road networks is shown in Table 2, in terms of average travel time and throughput. The performance of all the RL methods is the average of three training runs with different random seeds.

We can see that the conventional methods have poor performance, *i.e.*, FixedTime, SOTL, and MaxPressure, especially in the road network of Shenzhen. This is because in these real road networks, traffic is different and dynamic, and the conventional methods that heavily rely on predefined rules or assumptions about traffic can easily fail. Note that we use the implementations of these methods in CoLight (Wei et al. 2019b).

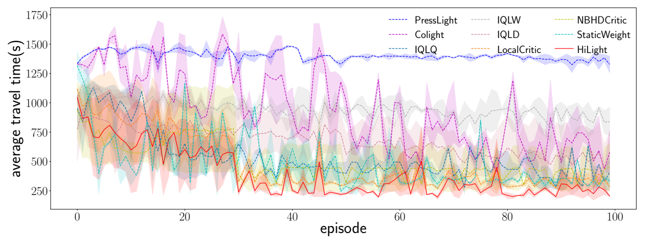
CoLight and PressLight are cooperative traffic signal control methods. PressLight optimizes the pressure that indirectly promotes the coordination between neighboring intersections. However, it does not perform well in some of the road networks, such as Manhattan and Shenzhen. The main reason is that PressLight may only work in arterial roads as shown in their paper (Wei et al. 2019a), where the experiments were all conducted in arterial roads. Moreover, from Figure 7, the convergence of PressLight in terms of average travel time is slow and unstable. This indicates that the optimization of pressure is not aligned with average travel time. CoLight optimizes queue length at each intersection by exploiting shared observations from neighboring intersections. However, CoLight does not perform well in the road network of Shenzhen. The reason may be that the irregular road network incurs the learning difficulty for CoLight. HiLight significantly outperforms all the baselines in all the road networks, and the learning of HiLight is much more quickly and stable than CoLight and PressLight as depicted in Figure 7. Unlike existing methods, HiLight learns to directly optimize average travel time locally by aligning the specialized sub-policies and additionally optimizes neighborhood travel time to encourage cooperation among agents.



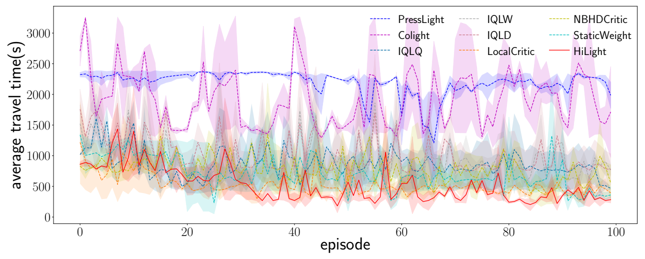
(a) Jinan



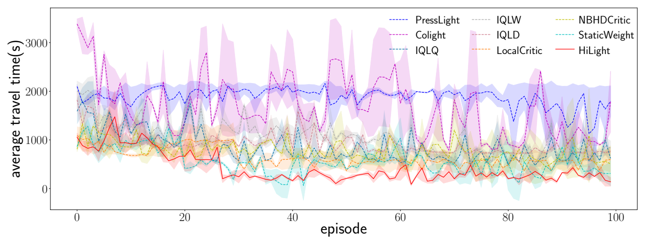
(b) Hangzhou



(c) Manhattan



(d) Shenzhen (D)



(e) Shenzhen (E)

Figure 7: Learning curves in the road network of Jinan, Hangzhou, Manhattan, and Shenzhen (D, E), plotted based on three training runs using different random seeds. As PressLight and CoLight take much more episodes to converge than HiLight, the curves are plotted with 100 episodes for clear presentation.

Ablation Studies

The learned sub-policies (*i.e.*, IQLQ, IQLW, amd IQLD) do not performance as well as LocalCritic. Among the sub-policies, IQLQ performs the best in all the road networks. Optimizing waiting time or delay alone cannot effectively reduce average travel time, while optimizing queue length is more stable and achieve better performance. This may be the reason that queue length is frequently chosen as the optimization objective in existing methods. The learning curve of IQLQ is even comparable with LocalCritic in the road networks of Jinan and Hangzhou. However, in the road network of Shenzhen, IQLQ performs much worse than LocalCritic, indicating that optimizing queue length alone is not a good option in irregular networks. Moreover, as these sub-policies optimize different targets, they indeed behave differently, as illustrated in Figure 6, in terms of percentage of

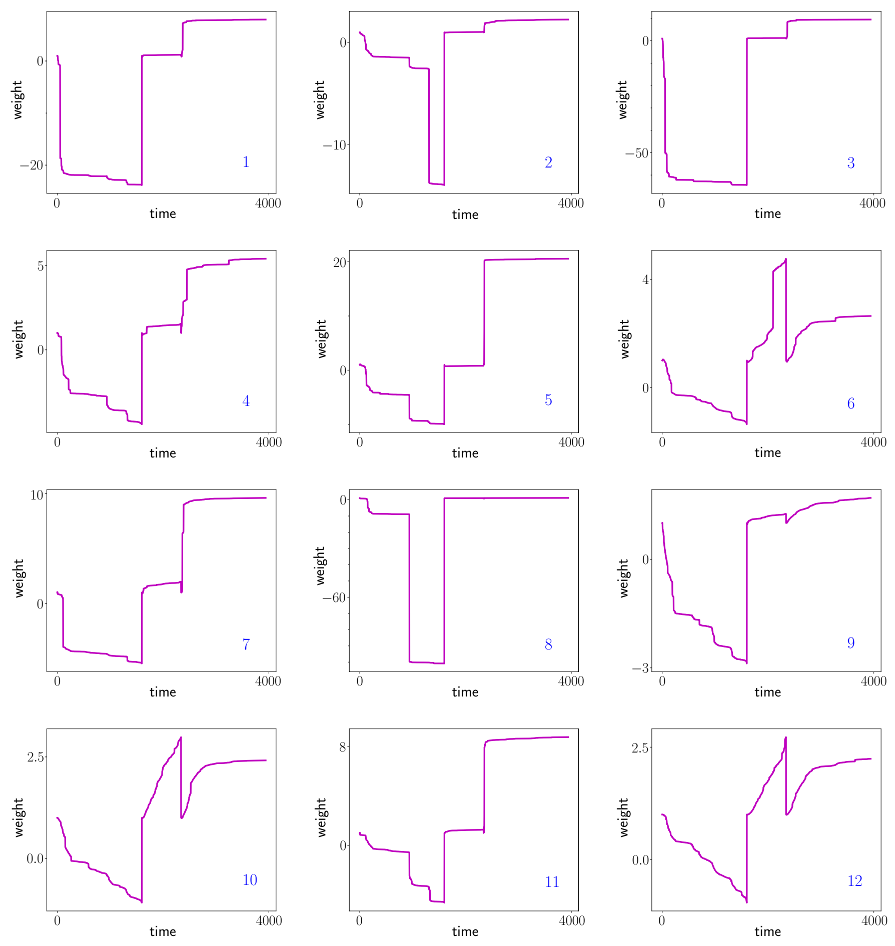


Figure 8: The change of the weight (w) of each intersection in the road network of Jinan during training. The number in each plot corresponds to the intersection index in Figure 5a.

time taken by each phase in an episode.

LocalCritic that employs a controller to optimize only local travel time by selecting among sub-policies outperforms its sub-policies, thought the difference is marginal in some road networks. However, the sub-policies provide the adaptability for LocalCritic in various kinds of road networks. NBHDCritic does not perform as well as LocalCritic, even worse than some of the sub-policies. This is because neighborhood travel time greatly depends on the policies of other agents in the neighborhood and thus it is hard to directly optimize by itself. StaticWeight that optimizes both objectives (*i.e.*, $w = 1$) outperforms LocalCritic and NBHDCritic. This is because taking into consideration neighborhood travel time while optimizing local travel time encourage the cooperation between neighboring agents. As the neighborhood of agents overlaps, this could boost cooperation across the entire network and thus reduce average travel time.

By adaptively weighting the importance of optimizing neighborhood travel time for optimizing local travel time at each intersection, the performance of HiLight further improves. The reason is that adaptive weighting can dynamically find a good weight that captures the importance of optimizing neighborhood travel time under different traffic patterns. Figure 8 shows the change of the weights of 12 intersections in the road network of Jinan during training. At each intersection the weight varies differently and converges to different values. This is because the intersections have various traffic patterns and the importance of optimizing neighborhood travel time also varies.

Conclusion

We have proposed HiLight, a hierarchical RL method for cooperative traffic signal control. The controller minimizes local travel time and neighborhood travel time jointly with adaptive weighting by selecting among the sub-policies that optimize short-term targets. Cooperation among agents is encouraged by the optimization of neighboring travel time. It is empirically demonstrated in four real datasets that HiLight significantly outperforms the existing RL methods for cooperative traffic signal control.

Acknowledgments

This work is supported by NSFC under grant 61872009.

References

Aslani, M.; Mesgari, M. S.; and Wiering, M. 2017. 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* 85: 732–752.

Bacon, P.-L.; Harb, J.; and Precup, D. 2017. The option-critic architecture. In *AAAI'17*.

Chen, C.; Wei, H.; Xu, N.; Zheng, G.; Yang, M.; Xiong, Y.; Xu, K.; and Li, Z. 2020. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *AAAI'20*.

Cools, S.-B.; Gershenson, C.; and D'Hooghe, B. 2013. Self-organizing traffic lights: A realistic simulation. In *Advances in applied self-organizing systems*, 45–55. Springer.

El-Tantawy, S.; and Abdulhai, B. 2012. Multi-agent reinforcement learning for integrated network of adaptive traffic signal controllers. In *ITSC'12*.

Frans, K.; Ho, J.; Chen, X.; Abbeel, P.; and Schulman, J. 2018. Meta learning shared hierarchies. In *ICLR'18*.

Jiang, J.; Dun, C.; Huang, T.; and Lu, Z. 2020. Graph Convolutional Reinforcement Learning. In *ICLR'20*.

Jiang, J.; and Lu, Z. 2019. Learning fairness in multi-agent systems. In *NeurIPS'19*.

Koonce, P.; and Rodegerdts, L. 2008. Traffic signal timing manual. Technical report, United States. Federal Highway Administration.

Liang, X.; Du, X.; Wang, G.; and Han, Z. 2018. Deep reinforcement learning for traffic light control in vehicular networks. *arXiv:1803.11115* .

Lin, X.; Baweja, H.; Kantor, G.; and Held, D. 2019. Adaptive auxiliary task weighting for reinforcement learning. In *NeurIPS'19*.

Mirchandani, P.; and Head, L. 2001. A real-time traffic signal control system: architecture, algorithms, and analysis. *Transportation Research Part C: Emerging Technologies* 9(6): 415–432.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540): 529–533.

Nachum, O.; Gu, S. S.; Lee, H.; and Levine, S. 2018. Data-efficient hierarchical reinforcement learning. In *NeurIPS'18*.

Nishi, T.; Otaki, K.; Hayakawa, K.; and Yoshimura, T. 2018. Traffic signal control based on reinforcement learning with graph convolutional neural nets. In *ITSC'18*.

Rashid, T.; Samvelyan, M.; De Witt, C. S.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML'18*.

Roess, R. P.; Prassas, E. S.; and McShane, W. R. 2004. *Traffic engineering*. Pearson/Prentice Hall.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv:1707.06347* .

Son, K.; Kim, D.; Kang, W. J.; Hostallero, D. E.; and Yi, Y. 2019. QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *ICML'19*.

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

Varaiya, P. 2013. The max-pressure controller for arbitrary networks of signalized intersections. In *Advances in Dynamic Network Modeling in Complex Transportation Systems*, 27–66. Springer.

Vezhnevets, A. S.; Osindero, S.; Schaul, T.; Heess, N.; Jaderberg, M.; Silver, D.; and Kavukcuoglu, K. 2017. Feudal networks for hierarchical reinforcement learning. In *ICML'17*.

Wei, H.; Chen, C.; Zheng, G.; Wu, K.; Gayah, V.; Xu, K.; and Li, Z. 2019a. PressLight: Learning max pressure control to coordinate traffic signals in arterial network. In *KDD'19*.

Wei, H.; Xu, N.; Zhang, H.; Zheng, G.; Zang, X.; Chen, C.; Zhang, W.; Zhu, Y.; Xu, K.; and Li, Z. 2019b. Colight: learning network-level cooperation for traffic signal control. In *CIKM'19*.

Wei, H.; Zheng, G.; Yao, H.; and Li, Z. 2018. IntelliLight: A reinforcement learning approach for intelligent traffic light control. In *KDD'18*.

Xu, N.; Zheng, G.; Xu, K.; Zhu, Y.; and Li, Z. 2019. Targeted knowledge transfer for learning traffic signal plans. In *PAKDD'19*.

Zang, X.; Yao, H.; Zheng, G.; Xu, N.; Xu, K.; and Li, Z. 2020. MetaLight: Value-based meta-reinforcement learning for traffic signal control. In *AAAI'20*.

Zhang, H.; Feng, S.; Liu, C.; Ding, Y.; Zhu, Y.; Zhou, Z.; Zhang, W.; Yu, Y.; Jin, H.; and Li, Z. 2019. Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In *WWW'19*.

Zheng, G.; Xiong, Y.; Zang, X.; Feng, J.; Wei, H.; Zhang, H.; Li, Y.; Xu, K.; and Li, Z. 2019. Learning phase competition for traffic signal control. In *CIKM'19*.