A EXPERIMENTS DETAILS

Appendix A encompasses the details of all numerical experiments. In Appendix A.1, we detail the results of model generalization. Appendices A.2 describes the model speed of TSC methods. Appendices A.3 shows the learning curve of typical TSC methods.

A.1 MODEL GENERALIZATION

Advanced-MP-JN1 is online-RL method, while BC-JN1, TD3+BC-JN1, BEAR-JN1, CQL-JN1, and OfflineLight-JN1 are offline methods, which all are only trained on offline dataset $JiNan^1$ in TSC-OID, which is around 20% of all data from TSC-OID. Then, these methods is directly deployed on other traffic topologies and flows. Intuitively pleasing but with little surprise, OfflineLight-JN1 has almost achieved a similar performance to the SOTA RL methods.

As shown in Table 2, OfflineLight-JN1 performs the best among all RL-Transfer methods on most real-world datasets, which are trained on offline dataset $JiNan^1$ in TSC-OID. Hence, partial offline training is essential for further decreasing the time and computation cost.

Type	Method	JiNan			HangZhou		New York	
		1	2	3	1	2	1	2
RL-Transfer	Advanced-MPLight-JN1	258.29	267.24	289.34	345.20	334.68	1253.62	1525.84
	BC-JN1	282.31	261.98	259.48	311.94	355.14	1152.74	1489.49
	TD3-BC-JN1	284.70	242.91	256.62	286.83	331.36	1201.09	1536.39
	BEAR-JN1	271.29	242.48	257.51	283.19	325.94	**1033.88**	**1425.65**
	CQL-JN1	265.68	241.86	240.52	283.29	325.71	1119.83	1465.77
	OfflineLight-JN1	**263.31**	**240.42**	**237.93**	**284.11**	**308.87**	1114.61	1450.01

Table 2: Performance comparison of different methods evaluated on JiNan, HangZhou and New York real-world datasets (ATT in seconds, the smaller the better).

In addition, the performance on transferability is denoted as a transfer ratio: $t_{transfer}/t_{train}$, where $t_{transfer}$ and t_{train} are the ATT performance of transfer and direct training, respectively. The smaller the transfer ratio is, the better the model's transferability will be. At last, we compare the performance of transferred and trained processes of the Advanced-MPLight and OfflineLight.

As shown in Figure 4, we can see that OfflineLight has shown an obvious advantage of transferability. Moreover, OfflineLight inherently has a vast generalization for its framework design.

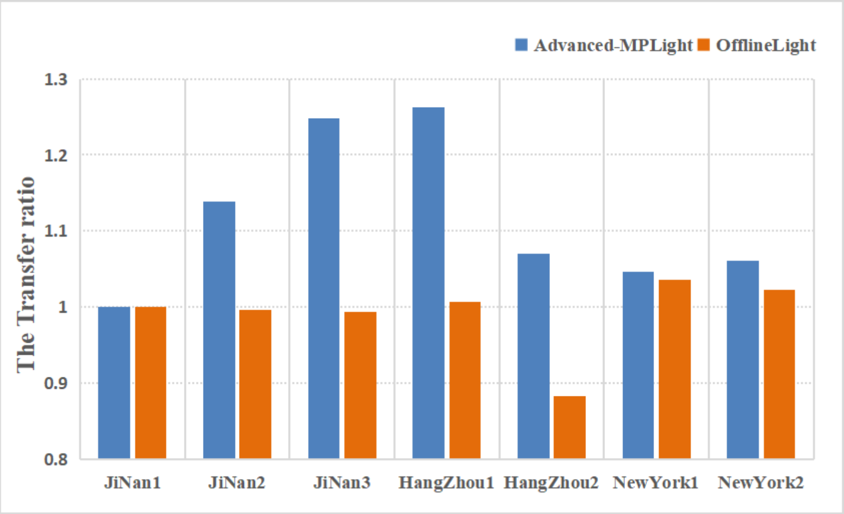


Figure 4: The illustration of model generalization by transfer ratio comparison.

A.2 MODEL SPEED

We compared the training time and running time of several classic online RL algorithms and OfflineLight, all operating on the same computing resources. Specifically, for training time, we measured the total time from the start of the training to its completion under the same configuration. For running time, we measured the average time required to transition from the input traffic state to the output signal action in a single event. OfflineLight leverages a large amount of pre-collected data for training, which helps improve data utilization efficiency during the learning process. In contrast, online RL methods typically require real-time interaction with the environment and involves a longer online exploration process.

As shown in Table 3, OfflineLight is $\mathbf{15.6x}$ times faster than Advanced-CoLight in $Jinan_1$ dataset, although their performances in experiments are close. In addition, OfflineLight has an approximate training time compared to CQL but shows the best performance in running time, which is more critical for real-world deployment.

Table 3: Comparison on training speed and running speed in milliseconds.

Method	Training Time	Running Time
FixedTime	0	57.42
FRAP	22988.91	121.94
MPLight	26902.15	135.55
CoLight	6166.31	155.22
Advanced-CoLight	7016.85	99.21
CQL	449.92	77.31
OfflineLight	502.37	62.60

A.3 LEARNING CURVE

We choose Advanced-CoLightZhang et al. (2022b), and our OfflineLight to compare the stability of the learning process. Each method is executed for a total of 30 episodes, with evaluations conducted every three episodes. In each evaluation, the average travel time is reported for a single episode, providing insights into the performance of the algorithms. These online reinforcement learning models demonstrate exceptional performance and robust stability in practical research applications.

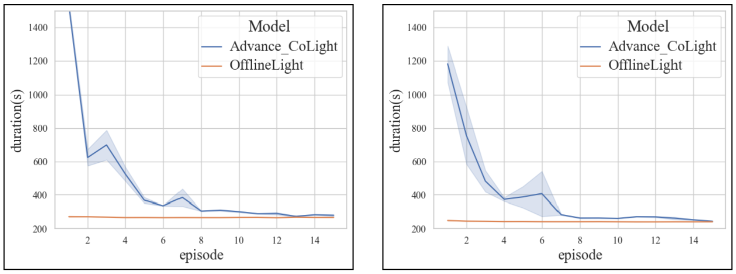
As illustrated in Figure 5, we observe that OfflineLight exhibits remarkably stable and consistently smooth learning curves across four real-world datasets. In contrast, the compared methods require a relatively longer and less stable learning process.

A.4 PERFORMANCE COMPARISON OF OFFLINELIGHT AGAINST CONVENTIONAL METHODS

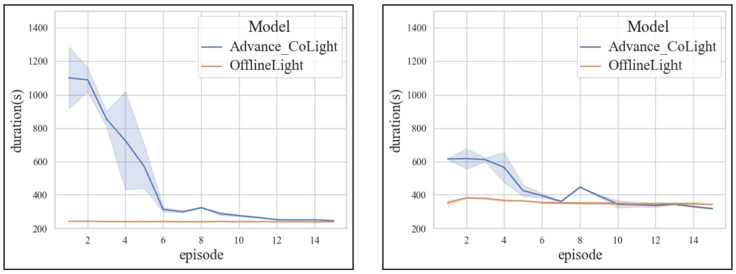
We added the performance comparison of OfflineLight against conventional methods, such as:Fixed-time, Max Pressure (MP). As shown in Table 4, our OfflineLight has evident advantages in ATT performance.

Type	Method	JiNan			HangZhou		New York
		1	2	3	1	2	1
Traditional Mehtods	Fixed-time	428.11	368.76	383.01	495.57	406.65	1507.12
	MP	273.96	245.38	243.80	288.54	348.98	1179.55

Table 4: Performance of traditional methods (ATT in seconds).



Jinan$_1$ **Jinan$_2$**



Jinan$_3$ **Hangzhou$_2$**

Figure 5: Comparison of learning curves in four datasets.

B METHODS DETAILS

In this section, we introduce the details of baseline methods, the specific settings of hyperparameters and network structures for OfflineLight.

B.1 DATASET

JiNan datasets: There are 12 (3×4) intersections on the road network. Each has a four-way intersection with two 800-meter-long (South-North) and two 400-meter-long (East-West) road segments. This traffic road network dataset includes three traffic flow datasets ($JiNan_1$,$JiNan_2$,$JiNan_3$).

HangZhou datasets: There are 16 (4×4) intersections on the road network. Each has a four-way intersection with two 800-meter-long (East-West) and two 600-meter-long (South-North) road segments. This traffic road network dataset includes two traffic flow datasets ($HangZhou_1$,$HangZhou_2$).

New York dataset: There are 192 (28×7) intersections on the road network. Each has two 300-meter (East-West) road segments and two 300-meter (South-North) road segments. This traffic road network dataset contains two traffic flow datasets ($Network_1$,$Network_2$).

Note: the New York dataset is much more extensive than JiNan and HangZhou, which has not been used to generate TSC-OID.

B.2 BASELINE METHODS

Traditional RL Methods:

FRAP Zheng et al. (2019): Modeling phase conflicts, FRAP achieve invariant symmetry for flipped, rotated, and other symmetric situations in traffic flow.

AttendLight Oroojlooy et al. (2020): Introducing the attention mechanism helps to handle general traffic conditions.

CoLight Wei et al. (2019b): Using graph attention network to realize intersection cooperation.

AttentionLight Zhang et al. (2022a): Using the queue length as state and reward, applying self-attention mechanism to obtain phase relationships.

Efficient-MPLight Wu et al. (2021): FRAP based model, using current phase and efficient traffic movement pressure as observation, intersection pressure as reward.

Advanced-CoLight Zhang et al. (2022b)): CoLight based model, using current phase, efficient running vehicle, efficient pressure as the state, queue length as reward. It is the current state-of-the-art method.

PRGLight Zhao et al. (2021): Combining traffic prediction and traffic light control to jointly control the traffic light phase and traffic light duration.

Behavior Cloning (BC)

Behavior Cloning (BC) Torabi et al. (2018): Capitalizing on the expertise of an agent's behavior, the behavioral cloning algorithm acquires the ability to replicate its actions by carefully studying the exhibited performance.

Offline RL Methods:

CQL Kumar et al. (2020): Mitigating overestimation bias by incorporating conservative constraints into value estimation, CQL leads to a more effective balance between exploration and exploitation during the learning process.

TD3+BC Fujimoto & Gu (2021): Leveraging the stability of TD3's value function learning with the expert demonstrations provided by BC, the algorithm leads to improved performance and sample efficiency in reinforcement learning tasks.

BEAR Kumar et al. (2019): Addressing the overestimation bias issue in off-policy learning, the algorithm employs bootstrapped ensembles of value functions to effectively estimate the uncertainty in value estimates.

Combo Yu et al. (2021): Regularizing the value function on out-of-support state-action tuples generated via rollouts under the learned model results in a conservative estimate of the value function for out-of-support state-action tuples, without requiring explicit uncertainty estimation.

In contrast, our OfflineLight is an Offline-AC based method, which constrains Q-function and policy simultaneously.

B.3 OFFLINE DATASETS GENERATION

In most existing works, it is still unclear which states and rewards are the best choices. We need to use a variety of states and rewards for the dataset to enhance the model's generalization. Moreover, the offline dataset is generated by different logging policies, similar to how the data buffer of off-policy RL (Q-learning or DDPGLillicrap et al. (2016)) is generated.

Table 5: Typical RL-related works with different states and rewards

RL-based Method	State	Reward
FRAP	NV	QL
AttendLight	NV-segments	QL
CoLight	NV	QL
AttentionLight	QL	QL
Efficient-MPLight	EP	TMP
Advanced-CoLight	EP and ERV	QL

The traffic states include the number of vehicles (NV), the number of vehicles under segmented roads (NV-segments), the queue length (QL), the efficient traffic movement pressure (EP), effective running vehicles (ERV), and the traffic movement pressure (TMP). The traffic reward consists of QL and TMP.

Hence, we record two groups (five offline datasets) for three epochs of training (denoted as $JiNan^1$, $JiNan^2$, $JiNan^3$, $HangZhou^1$, and $HangZhou^2$) by the training process of these RL methods that we choose. Each dataset records groups according to the traffic state and reward, as shown in

Table 5. The TSC-OID datasets are available online at anonymous Github[2] and the sizes of JiNan and Hangzhou offline dataset files are $843.9MB$ and $1130MB$, respectively.

B.4 NETWORK STRUCTURES

We introduce the structures of the Actor and Critic networks. All the networks use the same structure settings. The structure has three parts: phase embedding, lane embedding, and concat embeddings for the result. We first embed the phase feature in phase embedding and output it to a 4×4 dimensional matrix. Then, in lane embedding, we apply the MLP layers to embed the lane feature and output it to a 4×44 dimensional matrix. After that, we concat the embeddings of phase and lane features and use a 4-head MHA layer to learn the attention score between features. Finally, we employ an MLP layer to output the final result. Also, We summarize the network structures of OfflineLight as shown in Table 6.

Table 6: Network structure design

Layer name	Actor Critic (input size, output size)

# Phase embedding	Embedding(8, 8x4) 	
	Sigmoid(8x4, 8x4)
	Reshape(8x4, 2x4x4)
	Lambda(2x4x4, 4x4)

# Lane embedding	Reshape(12x11, 12x11x1) 	
	MLP(12x11x1, 12x11x4)
	Sigmoid(12x11x4, 12x11x4)
	Reshape(12x11x4, 12x44)
	Split(12x44, 1x44)
	Concat(1x44, 2x44)
	Lambda(2x44, 1x44)
	Concat(1x44, 4x44)

# Concat embeddings	Concat([4x44, 4x4], 4x48) 	
	MHA(4x48, 4x48)
	MLP(4x48, 4x20)
	Relu(4x20, 4x20)
	MLP(4x20, 4x20)
	Relu(4x20, 4x20)
	MLP(4x20, 4x1)
	Reshape(4x1, 4)

B.5 HYPERPARAMS OF OFFLINELIGHT

We detail the hyperparameters used in the experiment for OfflineLight. The same hyperparameters for both Actor and Critic networks: we set the multi-head attention counts to 4; we use a batch size

---

[2] https://anonymous.4open.science/r/OfflineLight-6665/README.md

Table 7: Hyperparameters of OfflineLight

Hyperparameter	Actor	Critic
batch size	1000	
MHA head counts	4	
optimizer	Adam	
learning rate	0.001	
τ	0.1	
discount(γ)	-	0.8
normally distributed noise std	-	0.02
clip noise c	-	0.2
update every t times	10	1

of 1000 for each step to train the network; we apply the Adam optimizer with a learning rate of 0.001 to update the parameters of the networks; we set τ to 0.1 to update the target networks.

The different hyperparameters for Actor and Critic networks: for the Critic network, we use discount γ of 0.8 to update the bellman function, and we add the normally distributed noise of 0.02 and clip it to the range -0.2 to 0.2 for better explore action space. For the Actor network, we update the network every ten times while we update the Critic network every one time. We summarize the hyperparameters of OfflineLight as shown in Table 7.

C ABLATION STUDY

C.1 CONSTRAINS FROM OFFLINELIGHT

we remove the constraints of actor and critic in OfflineLight respectively, forming OfflineLight-without-actor-constrain (OL-WAC), OfflineLight-without-critic-constrain (OL-WCC). Furthermore, we remove all constraints of actor and critic, leading to OfflineLight-without-both-constrain (OL-WBC), which is vanilla DDPGLillicrap et al. (2016). As shown in Table8, we trained these methods on $JiNan^1$ offline dataset and directly conducted on $JinNan_1$ traffic flow and topology. There is a significant decline in model performance when the actor and critical constraints are absent, underscoring the crucial role played by our AC framework. There is a significant decline in model performance when the actor and critical constraints are absent, underscoring the crucial role played by our AC framework.

Table 8: Ablation studies on constrains from OfflineLight

	Average waiting time on $JiNan_1$ (s)
OL-WAC	1284 ± 4.3
OL-WCC	1496 ± 0.0
OL-WBC	1275.83 ± 10.3

C.2 DISTANCE FUNCTION

We set the policy distance function between π_θ and the behavior policy π_β as $Dis(\pi_\theta, \pi_\beta)$. The choice of this distance function can be flexibly implemented. In OfflineLight, we apply Kullback–Leibler (KL) divergence $KL(\pi_\theta, \pi_\beta)$ to implement the distance function $Dis()$ for regularisation. Additionally, JS divergence, also known as JS distance, is a variant of KL divergence. we also can use JS divergence to implement the distance function $Dis()$ As shown in Table 9, we trained these methods on $JiNan^1$ and $HangZhou^1$ from offline dataset and directly conducted on $JinNan_1$ and $HangZhou_1$ traffic flow and topology, respectively. We can see that the performance of JS-div is a little bit degraded compared to KL-div, which can prove that using KL-div is more effective than JS for implementing our offline-AC framework.

Table 10: Ablation studies on distance function

	Average waiting time on $JiNan_1$ (s)	Average waiting time on $HangZhou_1$ (s)
KL-Div	265.32 ± 1.22	281.50 ± 0.93
JS-Div	263.31 ± 1.16	287.34 ± 0.00

D THEORETICAL PROOF

Actor: The design of the Actor combines KL divergence with a maximum Q value.

Critic: Below is the critic convergence proof. The objective function is

$$\begin{aligned}\hat{Q}^{k+1} \leftarrow \arg\min \alpha(\mathbb{E}_{s\sim\mathcal{D},a\sim(\pi_{\theta'}+\epsilon)}[Q'(s,a)]\\ -\mathbb{E}_{s,a\sim\mathcal{D}}[Q'(s,a)]) + \frac{1}{2}\mathbb{E}_{s,a\sim\mathcal{D}}[Q'(s,a) - \hat{\beta}^{\pi}\hat{Q}^{k}(s,a)]^2\end{aligned} \tag{6}$$

$$\begin{aligned}L(Q) = \alpha(\mathbb{E}_{s\sim\mathcal{D},a\sim(\pi_{\theta'}+\epsilon)}[Q'(s,a)] - \mathbb{E}_{s,a\sim\mathcal{D}}[Q'(s,a)])+\\ \frac{1}{2}\mathbb{E}_{s,a\sim\mathcal{D}}([Q'(s,a) - \hat{\beta}^{\pi}\hat{Q}^{k}(s,a)]^2)\end{aligned} \tag{7}$$

Let $\nabla_{Q'}L(Q') = 0$

$$\begin{aligned}\nabla_{Q'}L(Q') = -\sum_{s'}\hat{T}\left(s' \mid s,a\right)P(s,a)\left[\hat{\beta}^{\pi}Q^{k}\left(s',a'\right) - Q'(s,a)\right]\\ +\alpha d^{\pi}(s)(\pi_{\theta'}(a \mid s) - \pi(a))\end{aligned} \tag{8}$$

Sampling acquisition in the empirical state transfer distribution $\hat{T}\left(s_{t+1} \mid s_t, a_t\right)$ to get s_{t+1}, sampling from the marginal empirical distribution $\hat{d}^{\pi_{\theta'}}\left(s_t\right)$ to obtain s_t, sampling from the empirical distribution $\hat{\pi}_{\theta'}\left(a_t \mid s_t\right)$ for a_t.

$$\begin{aligned}\hat{d}^{\pi_{\theta'}}\left(s_t\right) &= \frac{\sum_{s\in D}1\left(s=s_t\right)}{|D|}\\ \hat{\pi}_{\theta'}\left(a_t \mid s_t\right) = \frac{P\left(s_t,a_t\right)}{\hat{d}^{\pi_{\theta'}}\left(s_t\right)} &= \frac{\sum_{s,a\in D}1\left(s=s_t,a=a_t\right)}{\sum_{s\in D}1\left(s=s_t\right)}\\ \hat{T}\left(s_{t+1} \mid s_t,a_t\right) = \frac{P\left(s_t,a_t,s_{t+1}\right)}{P\left(s_t,a_t\right)} &= \frac{\sum_{s,a,s'\in D}1\left(s=s_t,a=a_t,s'=s_{t+1}\right)}{\sum_{s,a\in D}1\left(s=s_t,a=a_t\right)}\end{aligned} \tag{9}$$

According to Equation 9, we can get

$$Q^{k+1}(s,a) = \hat{\beta}^{\pi}Q^{k}(s,a) - \alpha[\frac{\pi_{\theta'}(a \mid s)}{\pi(a \mid s)} - 1] \tag{10}$$

Lemma 1 *For a given distribution $\pi_{\theta'}(a|s) = \pi(a|s)$ with factor $\alpha > 0$, it holds that supp$(\pi_{\theta'}) \subset$ supp(π).*

Lemma 2 *If the policy gradient updates are very slow (at a sufficiently small update rate), disregarding sampling errors, that is, when $\hat{\beta}^{\pi} = \beta^{\pi}$, then selecting $\pi_{\theta'} = \hat{\pi}^{k}$ guarantees that at every step of the iterative update process, $\hat{V}^{k+1}(s) \le V^{k+1}(s)$.*

From Lemmas 1 and 2, it can be inferred that when the policy gradient updates are very slow (with minimal variation between $\hat{\pi}^{k+1}$ and $\hat{\pi}^{k}$), it ensures that:

$$\begin{aligned}E_{a\sim\hat{\pi}^{k+1}}[\hat{Q'}^{k+1}(s,a)] \le E_{a\sim\hat{\pi}^{k+1}}[\beta^{\pi}\hat{Q'}^{k}(s,a)]\\ \hat{V}^{k+1}(s) \le V^{k+1}(s)\end{aligned} \tag{11}$$