2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)
October 8-12, 2022, Macau, China

Meta-Reinforcement Learning for Centralized Multiple Intersections Traffic Signal Control

Yanyu Ren[1], Jia Wu[1*], Chenglin Yi[1], Yunchuan Ran[2], Yican Lou[1]

***Abstract*— Despite the success of reinforcement learning to traffic signal control, conventional reinforcement learning-based methods have mostly tackled the traffic signal control for a single network, which requires a large computational cost, and has insufficient generalization ability to the new environment. To solve the above problems, we propose a gradient-based meta-reinforcement learning method for the centralized traffic signal control to extract meta-knowledge from multiple meta-training tasks and utilize the accumulated meta-knowledge to a new task, thus improving the training efficiency. In addition, to mitigate the curse of dimensionality problem of the centralized control, we design a special agent, which decomposes the search space by using the Divide and Conquer paradigm. To the best of our knowledge, we are the first to combine the centralized control method with the meta-reinforcement learning method. Experimental results show that our method successfully generalizes to multiple unseen traffic networks. Compared with the traditional methods and the state-of-the-art reinforcement learning-based methods, our method achieves higher traffic signal control efficiency, faster convergence speed, and more stable performance.**

***Index Term*—Traffic signal control, Meta-reinforcement learning, LSTM model**

I. INTRODUCTION

With the rapid development of the city, the problem of traffic congestion is becoming worse, which seriously affects people's daily life. Optimizing traffic signal control (TSC) is an effective way to alleviate traffic congestion. The key problem in TSC is how to achieve global optimization in a road network, i.e., to ensure the optimal traffic signal control of all intersections, which has become a hot research topic in the field of intelligent transportation.

The traditional traffic control methods set a fixed time for each phase of the signal [1], or use a rule-based approach [2], which are not flexible to deal with dynamic traffic environment, and cannot achieve the cooperation between intersections. In recent years, more advanced methods have emerged, e.g. [13], [18], [19]. At the same time, reinforcement learning (RL), a learning framework that promotes agent to learn the optimal control strategy based on reward from the environment, has been applied to traffic signal control by many researchers [14] since its great potential in the field of TSC [3]. Researches of [4]–[6] have demonstrated that RL-based method is superior to traditional TSC methods. But RL-based TSC methods have a common issue: most of them can only deal with a specific traffic environment. For a new traffic environment, they must train from scratch and improve their performance through constant trial-and-error search, which is inefficient, even unacceptable in the real world when the time budget is limited.

To address this issue, many researches introduce meta-learning for TSC, which has rapid adaptability to new tasks. Meta-learning, also known as "learning to learn", intends to design models that can learn new skills or adapt to new environments rapidly with the accumulated knowledge from other tasks. MetaLight [7] combines meta-learning technology and a structure-agnostic model FRAP [12] to effectively control new intersection with different layout through prior knowledge. GeneraLight [8] uses meta-learning method to improve the generalization ability of RL-based traffic control methods in different traffic flow environment. Besides, it introduces the ideas of flow clustering, which aims to realize the adaptability of different traffic flows. However, the above models are only applied to single intersection for decentralized control. As is known to all, although the decentralized methods have been successfully applied in the large-scale network, but the cooperation of the agents is quite challenging since each agent makes decision by considering the close agents to achieve the global optimal policy. It is noted that the centralized control with a single agent theoretically can achieve the global optimal since it can observe the whole grid and make decision globally. However, as the scale of the network grows, the large state-action space makes it difficult for the agent to learn. How can we design a centralized TSC agent to alleviate the curse of dimensionality and to better generalize it over an unseen traffic network?

In this paper, we introduce meta-learning for the centralized TSC, where the goal is to learn a control agent that generalizes well over the different traffic networks (tasks), rather than a single environment, by using the accumulated meta-knowledge from previous tasks. Specifically, to achieve the centralized control, we design a special structure of the agent to decompose the state and action space by using Divide and Conquer paradigm to alleviate the curse of dimensionality problem. The proposed agent consists of an actor model and a critic model. The actor model utilizes the special properties of recurrent neural network to divide high-dimensional problems into multiple low-dimensional problems [16]. To enhance the generalization ability of the agent, we use gradient based meta-reinforcement learning (meta-RL) method to train the agent in various traffic networks

*This work was supported by the National Natural Science Foundation of China (Grant 61503059). Corresponding author: Jia Wu.

[1]Authors are with the School of Information and Software Engineering, University of Electronic Science and Technology of China, ChengDu, China. yanyuren@std.uestc.edu.cn; jiawu@uestc.edu.cn; chenglinyi@std.uestc.edu.cn; gengzhilou@gmail.com

[2]YunChuan Ran is Rice University, 6100 Min St, Houston, TX 77005. yunchuanran@rice.edu

2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC) | 978-1-6654-6880-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ITSC55140.2022.9922355

by learning the meta-knowledge of traffic control. For a new traffic network, our agent can fast adapt to it and deal with the traffic signal control more efficiently, thus greatly reducing the time of training.

In summary, our contributions are as follows:

- **Propose a method combining meta-RL and centralized TSC control:** by extracting common meta knowledge from multiple meta-tasks, our method can greatly enhance the generalization ability of the model. The TSC model can be applied to different traffic environments with fast adaptability. To the best of our knowledge, we are the first to apply meta-RL to a centralized control method.
- **Alleviate the curse of dimensionality problem of the centralized control:** we have specially designed the architecture of the agent, which utilizes LSTM model to decompose the high dimensional action space, thus mitigating the curse of dimensionality problem of the centralized control.
- **Conduct rich experiments to demonstrate the proposed method:** we have conducted a series of experiments on real-world datasets, including a traffic network with 196 intersection. As far as we know, the existing centralized control methods are difficult to deal with such a large-scale network. In addition, we have designed the ablation experiments to demonstrate the generalization ability of our model.

The rest of this paper is organized as follows. Section II formally defines the Meta-RL traffic control problem. Section III details our method. The experiment results is presented in Section IV. Finally, The conclusion of this paper is given in Section V.

II. PROBLEM DEFINITION

We first give the problem formulation of RL-based traffic grid signals control in the form of Markov Decision Process (MDP). Then, we define the problem of the meta-RL for traffic grid signals control.

A. RL-based Traffic grid signals control

An agent controls the traffic signals of N intersections, and selects the optimal phases according to the global state. The control process can be formulated as Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$, where:

1) System state space $\mathcal{S}$. $\boldsymbol{s}_t \in \mathcal{S}$ represents the agent's observation of the whole road network at time t, where $\boldsymbol{s}_t = [s_1^t, \ldots, s_N^t]$ and s_n^t ($n \in [1, N]$) is the state information of intersection i at time t, which includes the current phase at intersection i, the duration of current phase and the number of vehicles on each lane.
2) Action space of the agent $\mathcal{A}$. $\boldsymbol{a}_t = [a_1^t, \ldots, a_N^t]$ represents the joint action for the whole network, where a_n^t represents the decision for intersection n. The agent decides, every Δt period of time, a phase from its pre-defined phase set to be activated. It can be seen that the action space of the agent is 4^N if each intersection has 4 phases.
3) Transition probability $\mathcal{P}$ denotes the probability of transition to the next state, based on the current state $\boldsymbol{s}_t$ and $\boldsymbol{a}_t$, $\mathcal{P}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t) : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. Since the traffic environment is complex and dynamical, the transition probability is not easy to obtain.
4) Reward r is the evaluation of the action selected by the agent. The design of r is important since it can affect the final performance [17]. We design the reward as $\boldsymbol{r}_t = [r_1^t, \ldots, r_N^t]$, where r_n^t denotes the negative value of the sum of the queue lengths on the approaching lanes of intersection n at time t. The ultimate goal of the agent is to maximize the cumulative rewards.

B. Meta-RL for Traffic grid signals control

We consider a traffic network as a meta-task. Our goal is to fast adapt to new traffic network by learning the prior knowledge from $\mathfrak{T}_{meta} = \{\mathcal{T}_1, \ldots, \mathcal{T}_M\}$. To this end, we train a meta learner with parameters Θ_{meta} to extract meta-knowledge from meta-training tasks:

$$\begin{aligned} \Theta_{meta} &= \arg\min_{\Theta} \tfrac{1}{M} \textstyle\sum_{i=1}^{M} \mathcal{L}(\Theta_i, \mathcal{T}_i) \\ \Theta_i &= f_\Theta(\mathcal{T}_i) \end{aligned} \tag{1}$$

where $\mathcal{L}(\cdot)$ is a meta-training loss which is related to the expected rewards and Θ_i is the parameter of policy for solving task $\mathcal{T}_i$ and generated by $f_\Theta(\cdot)$. $f_\Theta(\cdot)$ is defined based on particular meta-RL algorithm.

III. METHOD

In this section, we will introduce the main components of the agent to achieve the centralized control and how to meta-train the agent.

A. Agent's Network Structure

As we know, the major challenge of the centralized control is the large search space, especially the action space. For example, for a road network with N intersections where the traffic signal at each intersection has four phases, the joint action space is up to 4^N. To alleviate the curse of dimensionality problem, we designed a novel agent, which is inspired by the idea of Divide and Conquer paradigm. The internal structure of our agent is described in Figure 1, which is composed of two parts: an actor model π_θ (a policy learner) and a critic model V_ϕ (a value function). The policy learner makes decisions conditioned on the current state and the value function evaluates the current state, the expected return if the agent starts in that state, and then acts according to a particular policy forever after. To decompose the state and the action space, we specially designed the actor model whose core component is a Long-Short Term Memory (LSTM) network and the critic model is a MLP network.

The agent works as follows (as shown in Fig. 1): at each time step t, the agent receives the state $\boldsymbol{s}_t$ from the environment. The actor unrolls N times: at each unrolling step $n \in [1, N]$, the actor makes action a_n^t for intersection n, based on s_n^t, the state of intersection n, and the previous decisions $\boldsymbol{a}_{1:n-1}^t = [a_1^t, \ldots, a_{n-1}^t]$. After N steps, the actor outputs the joint action $\boldsymbol{a}_t = [a_1^t, \ldots, a_N^t]$. The critic

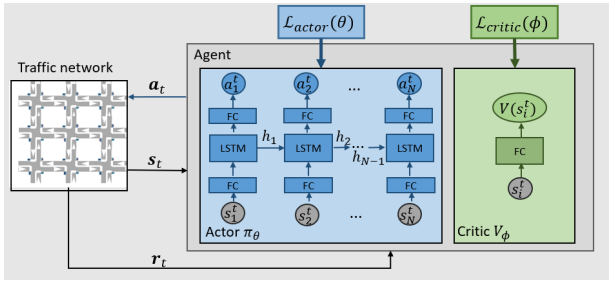

Fig. 1. Internal architecture of agent. Our agent is composed of two parts: an actor model π_θ and a critic model V_ϕ. The actor model is mainly composed of LSTM network to sequentially output actions for each signal. For a grid with N signals, the agent receives, at each simulation time t, the state of the whole grid $\boldsymbol{s}_t = [s_1^t, \ldots, s_N^t]$; the actor model unrolls N steps to sequentially output decisions for each intersection. After N unrolling-steps, it combines all the sub-decisions to a joint action $\boldsymbol{a}_t = [a_1^t, \ldots, a_N^t]$. The critical model, composed of fully-connected network, evaluates the sub-state and helps to train the actor.

calculates the value function according to the current state and helps update the parameters of the actor with the reward emitted by the environment.

We can see from the above workflow that since the agent makes the joint action sequentially, it makes one decision for an intersection at each unrolling step, based on previous decisions, the action space is greatly reduced and it is much easier for the agent to search a good action at each unrolling step. In addition, since we use the LSTM neural network to construct the core part of the actor, it can remember the past and make decisions based on what it has learnt from the past. As authors in [20] claimed: any complex high-dimensional action can be selected incrementally, component by component, where each component probability also depends on components already selected earlier.

We use one of the modern Actor-Critic algorithms (AC), Proximal Policy Optimization (PPO-Clip) algorithm [11] to train our agent. Considering the characteristics of traffic signal control problem and the special design of our agent, we propose some tricks to facilitate the training process.

First, calculate the advantage function $A\left(s_n^t, a_n^t\right)$ for intersection n, which describes how much better it is to take a specific action a_n^t at the state s_n^t:

$$\begin{aligned} \delta_n^t &= r_n^t + \gamma V_\phi(s_n^{t+1}) - V_\phi(s_n^t) \\ A\left(s_n^t, a_n^t\right) &= \textstyle\sum_{t'=t}^{T} (\gamma\lambda)^{t'-t} \delta_n^t \end{aligned} \tag{2}$$

where $V_\phi(\cdot)$ is the critic model; λ and γ are discounted factors which adjust bias-variance tradeoff, where $0 \leq \lambda \leq 1$ and $0 \leq \gamma \leq 1$.

The loss of the actor model π_θ is then calculated by:

$$\begin{aligned} \mathcal{L}_{actor}(\theta) = & \frac{1}{T \times N} \sum_{t=1}^{T} \sum_{n=1}^{N} \min \left(pr_n^t(\theta) A\left(s_n^t, a_n^t\right) , \right. \\ & \left. \operatorname{clip}\left(pr_n^t(\theta), 1-\epsilon, 1+\epsilon\right) A\left(s_n^t, a_n^t\right) \right) \end{aligned} \tag{3}$$

where $\epsilon = 0.2$ and $pr_n^t(\theta)$ evaluates the differences of new and old policies, which is defined as follows:

$$pr_n^t(\theta) = \frac{\pi_\theta\left(a_n^t \mid s_n^t\right)}{\pi_{\theta_{\text{old}}}\left(a_n^t \mid s_n^t\right)} \tag{4}$$

where $\pi_\theta\left(a_n^t \mid s_n^t\right)$ represents the actions probability that the agent decides based on the state.

The loss of the critic model is defined as:

$$\mathcal{L}_{critic}(\phi) = \frac{1}{T \times N} \min \sum_{t=1}^{T} \sum_{n=1}^{N} (G_n^t - V_\phi(s_n^t))^2 \tag{5}$$

where G_n^t is the reward-to-go for the agent. Generally, $G_n^t \doteq \sum_{t'=t}^{T} \gamma^{t'-t} r_n^t$. However, the general setting of G_n^t has such a problem: when reaching the last simulation time step T, G_N^t will be much smaller than other G_n^t, thus leading to great variance during training. Therefore, we design G_n^t as follows:

$$G_n^t \doteq \sum_{t'=t}^{T} \gamma^{t'-t} r_n^t + \gamma^{T-t+1} V_\phi(s_n^{T+1}) \tag{6}$$

Where $V_\phi(s_n^{T+1})$ evaluates the state at $T+1$. Then, the parameters of the actor model and the critic model are updated by gradient descent.

B. Meta-Reinforcement Learning

We consider introducing meta-RL into the training procedure of the agent. Specifically, we aim to learn a meta parameter $\Theta_{meta} = \{\theta_{meta}, \phi_{meta}\}$, including parameters for the actor and the critic models, which contain the prior knowledge from the meta-training tasks to make the agent rapidly adapt to new task. The Meta-training procedure is divided into two phases: adaptation and meta-optimization (see Fig. 2):

Adaptation: Firstly, sample a batch of tasks $\mathcal{T}_{1:M}$ as meta-training tasks. For each task $\mathcal{T}_i$, initialize the agent with meta-parameters Θ_{meta}, i.e., $\Theta_i = \Theta_{meta}$ (each task corresponds to one agent). Then, train the agent along to adapt to the task. Specifically, at each episode, the agent Θ_i implements the task $\mathcal{T}_i$ for P steps and collect P samples $\mathcal{D}_{adapt}^i = \boldsymbol{c}_{1:P}$, where $\boldsymbol{c}_j = \{\boldsymbol{s}_j, \boldsymbol{a}_j, \boldsymbol{r}_j\}$, $j \in [1, P]$. Next, update parameters Θ_i of the agent with respect to the training samples in $\mathcal{D}_{adapt}^i$ based on the following loss function:

$$\Theta_i = \Theta_{meta} + \alpha \nabla_{\Theta_{meta}} \mathcal{L}(\Theta_{meta}, \mathcal{D}_{adapt}^i) \tag{7}$$

where $\mathcal{L}$ corresponds to the loss functions (3) and (5), i.e., $\mathcal{L}_{critic}$ and $\mathcal{L}_{actor}$. And α is the learning rate. Here, equation (7) corresponds to multiple gradient updates.

Meta-optimization: The goal of the meta-training procedure is to obtain meta-parameters with generalization ability on new tasks. We perform optimization on the meta-parameters after adaptation phase by learning the common meta-knowledge from multiple tasks. Specifically, for each task $\mathcal{T}_i$, the agent performs the task in Q steps, with the parameters updated in the adaptation phase, and collects Q samples in $\mathcal{D}_{meta}^i = \boldsymbol{c}_{1:Q}$. Then, update the meta-parameters by evaluating the performances of all tasks:

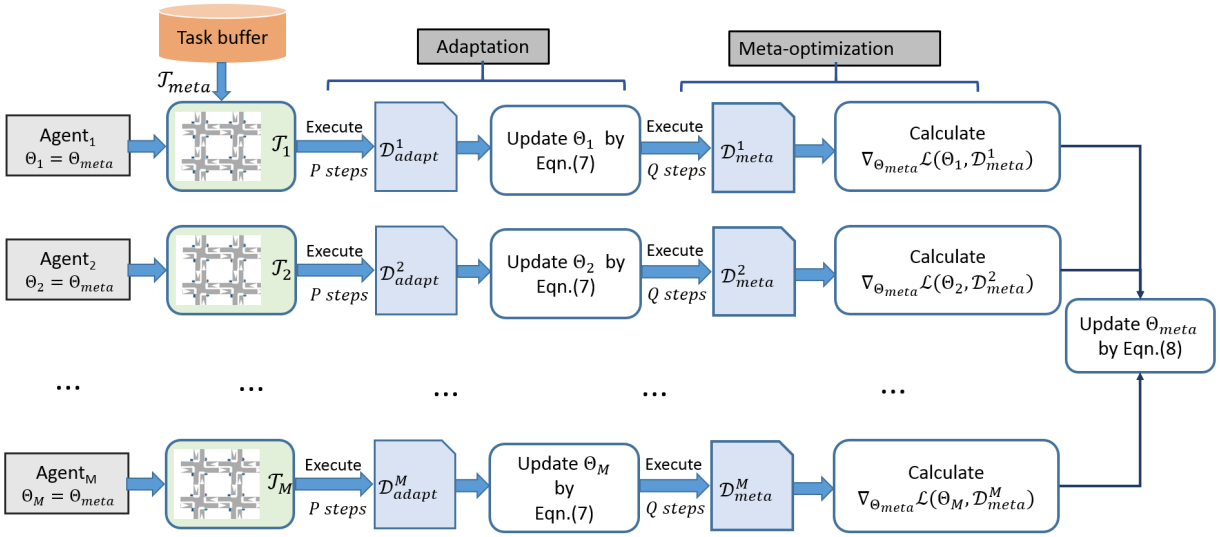

Fig. 2. Framework of meta-training process. The training process is divided into two phases: adaptation and meta-optimization. The meta-training begins by sampling a batch of M meta-training tasks from Task Buffer. At the adaptation phase, for each task $\mathcal{T}_i$, initialize $Agent_i$ with $\Theta_i = \Theta_{meta}$. Then, $Agent_i$ executes P steps and uses the collected samples in $\mathcal{D}^i_{adapt}$ to update parameters Θ_i by Eqn.(7). At the meta- optimization phase, $Agent_i$ with the updated parameters Θ_i executes Q steps, and uses the new samples in $\mathcal{D}^i_{meta}$ to calculate the gradient $\nabla_{\Theta_{meta}}\mathcal{L}(\Theta_i, \mathcal{D}^i_{meta})$. At last, meta-parameters Θ_{meta} is updated by the gradients from M meta-training tasks. The two phases are repeated until the end of the episode. After several iterations, the prior knowledge from meta-training tasks is learned and stored in meta-parameters Θ_{meta}.

$$\Theta_{meta} \longleftarrow \Theta_{meta} + \beta \sum_{m=1}^{M} \nabla_{\Theta_{meta}}\mathcal{L}(\Theta_i, \mathcal{D}^i_{meta}) \quad (8)$$

where $\mathcal{L}$ still corresponds to the equations (3) and (5) and β is the learning rate.

We can see that the gradient update in the equation (8) requires higher-order derivatives:

$$\begin{aligned} & \nabla_{\Theta_{meta}}\mathcal{L}(\Theta_i, \mathcal{D}^i_{meta}) \\ = \; & \nabla_{\Theta_{meta}}\mathcal{L}(\Theta_{meta} + \alpha\nabla_{\Theta_{meta}}\mathcal{L}(\Theta_{meta}, \mathcal{D}^i_{adapt}), \mathcal{D}^i_{meta}) \end{aligned} \quad (9)$$

To accelerate the training process, we use a first-order approximation :

$$\begin{aligned} \nabla_{\Theta_{meta}}\mathcal{L}(\Theta_i, \mathcal{D}^i_{meta}) &= \frac{\partial\mathcal{L}(\Theta_i, \mathcal{D}^i_{meta})}{\partial\Theta_{meta}} \\ &= \frac{\partial\mathcal{L}(\Theta_i, \mathcal{D}^i_{meta})}{\partial\Theta_i} \cdot \frac{\partial\Theta_i}{\partial\Theta_{meta}} \\ &\approx \frac{\partial\mathcal{L}(\Theta_i, \mathcal{D}^i_{meta})}{\partial\Theta_i} \\ &\approx \nabla_{\Theta_i}\mathcal{L}(\Theta_i, \mathcal{D}^i_{meta}) \end{aligned} \quad (10)$$

Through the above methods, the computational burden can be effectively reduced. The adaptation and the meta-optimization phases repeats alternatively in every episode. After several episodes, we get the meta-parameters Θ_{meta}.

Meta-testing: when the new task comes, initialize the parameters of the agent with Θ_{meta} to accelerate the training process. It is worth noting that Θ_{meta} may not be the best parameter of new tasks, but the empirical initialization provides a good starting point for solving new tasks since the meta-training procedure transfers prior knowledge from similar tasks.

IV. EXPERIMENT

In this section, we conduct extensive experiments to validate the effectiveness of our model. First, we compare our model with conventional TSC methods, RL-based TSC methods and meta-RL-based TSC methods on four real-world traffic networks. Second, we ablate the meta-learning component of our approach to better understand the salient features of our method.

A. Settings

We have carried out experiments on the new traffic simulation platform CityFlow [6], which is very flexible and supports the simulation of the large-scale road network control. Specifically, it interacts with agent as an environment, where it inputs and executes the action selected by agent, outputs state and reward.

B. Dataset

Our datasets are from [6], [12], [15]. Our meta-training tasks consist of a 6×6 synthetic dataset and a 3×4 real-world dataset. The meta-testing tasks are two small-scale road networks (4×4 Hangzhou road network, 3×4 Jinan road network), a medum-scale road network (16×3 NewYork), and a large-scale road network (28×7 NewYork). Their road network was imported from openstreetmap[1]. All the meta-data (road networks and traffic flows) are selected from the public datasets[2].

[1]https://www.openstreetmap.org
[2]https://traffic-signal-control.github.io/

C. Evaluation Metric

We use the travel time, which calculates the average travel time of all vehicles in the road network, as the evaluation metric since previous work has proved that travel time is one of the most popular and excellent evaluation Metric. It represents the average travel time of all vehicles from origin to destination (in seconds), which intuitively reflects the quality of the traffic signal control.

D. Compared Methods

We compared with the state-of-arts methods, including three conventional TSC methods, RL-based and meta-RL based TSC methods:

- **Fixed-time Control (FT) [1].** Its phase duration and phase switching sequence are designed in advance.
- **MaxPressure [2].** It is a more advanced traditional control method, which greedily selects the phase that maximizes the pressure.
- **Individual RL [3].** A decentralized control method where each agent controls one intersection, but there is no communication between agents. Individual RL doesn't use the parameter sharing techniques, i.e., it is hard to run the experiment on the NewYork 28×7 grid.
- **PressLight [5].** It combines MaxPressure with RL.
- **CoLight [6].** A decentralized control method, which uses graph attention network to realize the communication between agents.
- **MetaLight [7].** Combines MAML and Frap. However, the model is only applicable to single intersection. In order to apply it in the road network, we combine it with PressLight.
- **GeneraLight [8].** The model combines MAML and the ideas of flow clustering, and WGAN is used to generate different traffic flow data.

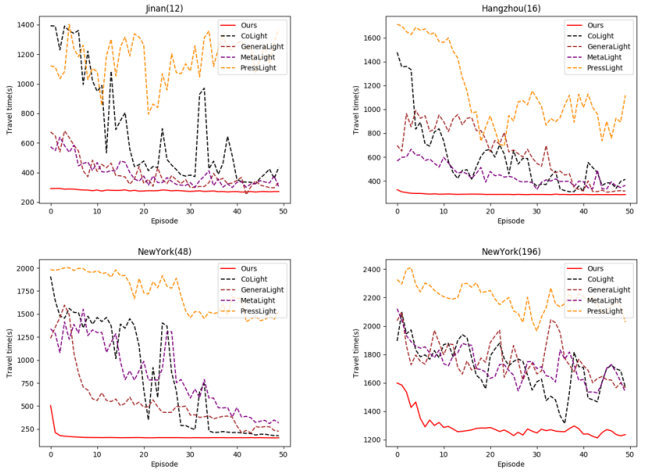

Fig. 3. Learning curves of all methods in datasets: D_{Jinan}(3x4), $D_{Hangzhou}$(4x4), $D_{Newyork}$(16x3), $D_{Newyork}$(28x7)

E. Performance Comparison

Table. I and Fig. 3 show the performance of our method, compared to the above baseline, from which we can see that:

1) **Our method** has achieved the optimal travel time, which demonstrates the effectiveness of our model. Besides, it can be seen from the learning curve during the training procedure (Fig.3) that our method has better convergence rate, which indicates the advantages of our model in terms of generalization and scalability.
2) **Conventional method** have no training process, so their performance is relatively stable. Due to the adaptability of pressure principle, the performance of MaxPressure is better than FT.
3) **RL-based methods:** 1) Individual RL: It performs well in Jinan and Hangzhou. Since the roadnet-scale of theses two datasets is small, and each intersection in the small road network can make a better decision based on their own information. While on the more large road network, e.g., NewYork, this model becomes difficult to train. 2) PressLight: It can be seen that the performance on the small road network is better than that on the large road network. But generally speaking, its control effect is not of high-quality. 3) CoLight: The performance is significantly better than other RL-based baselines because it uses GAT to learn the importance of neighbors.
4) **Meta-RL based methods:** On the whole, meta-RL based method is better than the common RL-based methods 1) MetaLight is applied to single intersection. We combine it with presslight. In this case, it has no cooperation mechanism , thus its performance is inferior to CoLight. 2) GeneraLight is built on MaxPressure, so it is similar to MaxPressure. As the scale of the road network increases, the performance becomes worse. In addition, the purpose of GeneraLight is to improve the generality of the model in different traffic flows.

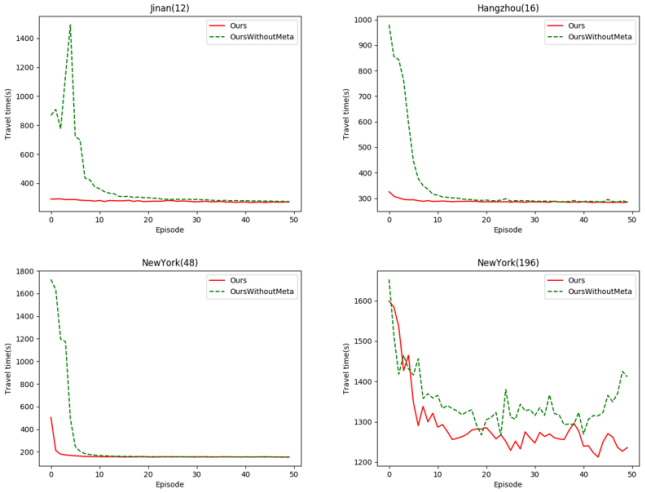

Fig. 4. Comparison of learning curves of our method with and without meta-learning

TABLE I

COMPARISON BETWEEN OUR METHOD AND TRADITIONAL AND RL-BASED METHODS ON REAL-WORLD DATA W.R.T AVERAGE TRAVEL TIME.

Models	Datasets			
	Jinan 3×4	Hangzhou 4×4	Newyork 16×3	Newyork 28×7
Fixed-time	814.11	718.29 -	1842.19	1943.69
Maxpressure	343.90	407.17	392.31	1611.08
Individual RL	459.43	489.87	1286.92	-
PressLight	689.87	754.09	1417.68	2021.59
CoLight*	291.14	297.26	161.13	1459.28
MetaLight.	298.91	331.17	308.32	1530.12
GeneraLight	293.29	305.66	207.14	1564.90
Ours	**261.36**	**283.11**	**152.33**	**1212.46**

*It is noted that the travel time data of CoLight is from [6]

F. Ablation Study

In order to prove the effectiveness of the meta-learning, we remove the meta learning process on our model, and compare our method with and without the meta-learning. Fig. 4 shows the performance comparison, from which we can clearly see that: compared with the model without meta-learning, the convergence speed of the model with meta-learning is greatly improved. This demonstrates the generalization and scalability of our model, i.e., the ability of rapid adaptation. Besides, we also note that the travel time that our model using meta-learning achieves is better than the one without meta-learning, which shows that the meta-knowledge we learned can help the agent find better control policy.

V. CONCLUSIONS

In this paper, we combine gradient based meta-learning and centralized control RL method to make our model have good generalization and scalability. The architecture of agent is specially designed, which uses LSTM network to make the incremental action selection, so as to mitigate the curse of dimensionality problem inherent in the centralized control. The agent is trained by the meta-learning procedure, which consists of adaptation and meta-optimization phases, to learn the meta-knowledge from meta-training tasks. Extensive experiments have been conducted on real-world datasets to demonstrate the effectiveness of our model. In addition, ablation experiments are conducted to verify the effectiveness of meta-learning. For the future work, we consider improve the meta-learning methods to reduce the excessive computational load of gradient based methods.

REFERENCES

[1] Miller, Alan J. "Settings for fixed-cycle traffic signals." Journal of the Operational Research Society 14.4 (1963): 373-386.

[2] Varaiya, Pravin. "Max pressure control of a network of signalized intersections." Transportation Research Part C: Emerging Technologies 36 (2013): 177-195.

[3] Wei, Hua, et al. "Intellilight: A reinforcement learning approach for intelligent traffic light control." Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018.

[4] Arel, Itamar, et al. "Reinforcement learning-based multi-agent system for network traffic signal control." IET Intelligent Transport Systems 4.2 (2010): 128-135.

[5] Wei, Hua, et al. "Presslight: Learning max pressure control to coordinate traffic signals in arterial network." Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2019.

[6] Wei, Hua, et al. "Colight: Learning network-level cooperation for traffic signal control." Proceedings of the 28th ACM International Conference on Information and Knowledge Management. 2019.

[7] Zang, Xinshi, et al. "Metalight: Value-based meta-reinforcement learning for traffic signal control." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34. No. 01. 2020.

[8] Zhang, Huichu, et al. "Generalight: Improving environment generalization of traffic signal control via meta reinforcement learning." Proceedings of the 29th ACM International Conference on Information & Knowledge Management. 2020.

[9] Finn, Chelsea, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks." International conference on machine learning. PMLR, 2017.

[10] Choe, Chung-Jae, et al. "Deep q learning with LSTM for traffic light control." 2018 24th Asia-Pacific Conference on Communications (APCC). IEEE, 2018.

[11] Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

[12] Zheng, Guanjie, et al. "Learning phase competition for traffic signal control." Proceedings of the 28th ACM International Conference on Information and Knowledge Management. 2019.

[13] Liu, Yong, et al. "Multi-agent game abstraction via graph attention neural network." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34. No. 05. 2020.

[14] Haydari, Ammar, and Yasin Yilmaz. "Deep reinforcement learning for intelligent transportation systems: A survey." IEEE Transactions on Intelligent Transportation Systems (2020).

[15] Wei, Hua, et al. "A survey on traffic signal control methods." arXiv preprint arXiv:1904.08117 (2019).

[16] Schmidhuber, Juergen. "Reinforcement Learning Upside Down: Don't Predict Rewards–Just Map Them to Actions." arXiv preprint arXiv:1912.02875 (2019).

[17] Silver, David, et al. "Reward is enough." Artificial Intelligence 299 (2021): 103535.

[18] Abdoos, Monireh, and Ana LC Bazzan. "Hierarchical traffic signal optimization using reinforcement learning and traffic prediction with long-short term memory." Expert systems with applications 171 (2021): 114580.

[19] Devailly, François-Xavier, Denis Larocque, and Laurent Charlin. "IG-RL: Inductive graph reinforcement learning for massive-scale traffic signal control." IEEE Transactions on Intelligent Transportation Systems (2021).

[20] Juergen Schmidhuber."Reinforcement Learning Upside Down: Don't Predict Rewards – Just Map Them to Actions." arXiv preprint arXiv:1912.02875 (2020).