Centralized Cooperation for Connected and Automated Vehicles at Intersections by Proximal Policy Optimization

Yang Guan, Yangang Ren, Shengbo Eben Li, *Senior Member, IEEE*, Qi Sun, Laiquan Luo, and Keqiang Li

***Abstract*—Connected vehicles will change the modes of future transportation management and organization, especially at an intersection without traffic light. Centralized coordination methods globally coordinate vehicles approaching the intersection from all sections by considering their states altogether. However, they need substantial computation resources since they own a centralized controller to optimize the trajectories for all approaching vehicles in real-time. In this paper, we propose a centralized coordination scheme of automated vehicles at an intersection without traffic signals using reinforcement learning (RL) to address low computation efficiency suffered by current centralized coordination methods. We first propose an RL training algorithm, model accelerated proximal policy optimization (MA-PPO), which incorporates a prior model into proximal policy optimization (PPO) algorithm to accelerate the learning process in terms of sample efficiency. Then we present the design of state, action and reward to formulate centralized coordination as an RL problem. Finally, we train a coordinate policy in a simulation setting and compare computing time and traffic efficiency with a coordination scheme based on model predictive control (MPC) method. Results show that our method spends only 1/400 of the computing time of MPC and increase the efficiency of the intersection by 4.5 times.**

***Index Terms*—Centralized coordination method, connected and automated vehicle, reinforcement learning, traffic intersection.**

I. Introduction

THE increasing demand for mobility poses great challenges to road transport. The connected and automated vehicles are attracting extensive attention, due to its potential to benefit traffic safety, efficiency and economy [1]–[3]. The widely studied, but also simplified, version of connected vehicle cooperation is the platoon control system on the highway. Platoon control aims to ensure that a group of connected vehicles in the same lane move at a harmonized longitudinal speed while maintaining desired inter-vehicle spaces [4]–[7]. As a typical scenario in urban areas, the intersection is more complex and challenging for multi-vehicle coordination than that on the highway. At the intersection, vehicles enter from different intersection entrances, cross their specific trajectories at the intersection zone, and leave the intersection at different exits. The complex conflict relationship between vehicles results in complicated vehicle decisions to avoid collisions at the intersection, which needs complicated design to guarantee traffic safety while improving traffic efficiency. Traffic signal control is commonly considered as the most effective method to resolve multi-vehicle coordination at intersections, and various strategies for urban traffic management have been developed. Goodall *et al.* developed a decentralized fully adaptive traffic control algorithm to optimize traffic signal timing [8]. Feng *et al.* presented a real-time adaptive signal phase allocation algorithm using connected vehicle data, which optimizes the phase sequence and duration by solving a two-level optimization problem [9]. These signal control strategies can only partially improve the traffic flow if all approaches to the intersection are not equally congested, and they cannot eliminate the stop delay of vehicles at intersections. Recently, several studies have started to focus on methods without using traffic signals for intersection coordination.

Currently, most of the existing studies without the traffic signal focused on centralized coordination methods which utilize the global information of the whole intersection to centralizedly organize the motion of all approaching vehicles. Lee and Park proposed a system that enables cooperation between automated vehicles and infrastructures for effective intersection operations [10]. The system tries to avoid the presence of any pair of conflicting vehicles in the intersection area at the same time by minimizing potential overlaps of trajectories of vehicles coming from all conflicting approaches at the intersection, then seeks a safe maneuver for each vehicle approaching the intersection and manipulates each of them. To construct the objective function of the nonlinear constrained optimization problem, they predicted the trajectory of each vehicle by fixing an acceleration from its current position to the end of the intersection. In their optimization problem, the computation requirements usually increase rapidly with the growth of the number of vehicles and should be determined carefully to ensure that the feasibility of the real-time implementation is guaranteed.

Manuscript received January 29, 2020; revised May 13, 2020 and August 15, 2020; accepted August 26, 2020. Date of publication September 23, 2020; date of current version November 12, 2020. This work was supported in part by the International Science and Technology Cooperation Program of China under Grant 2019YFE0100200, and in part by the Tsinghua University-Toyota Joint Research Center for AI Technology of Automated Vehicle. The review of this article was coordinated by Prof. X. Hu. *(Corresponding author: Shengbo Eben Li.)*

The authors are with the State Key Lab of Automotive Safety and Energy, School of Vehicle and Mobility, and with the Center for Intelligent Connected Vehicles and Transportation, Tsinghua University, Beijing 100084, China (e-mail: guany17@mails.tsinghua.edu.cn; ryg18@mails.tsinghua.edu.cn; lisb04@gmail.com; qisun@mail.tsinghua.edu.cn; luolaiquan@gmail.com; likq@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TVT.2020.3026111

Kamal *et al.* proposed a vehicle-intersection coordination scheme (VICS) without using any traffic light [11]. The scheme efficiently utilizes the intersection area by preventing each pair of conflicting vehicles from approaching their cross-collision point (CCP) at the same time, instead of reserving the whole intersection area for the conflicting vehicles successively. In this scheme, a risk function is proposed to quantify the risk of a collision of a pair of vehicles around their CCP. If two conflicting vehicles are very close to their CCP, the risk function returns a high value, and if at least one vehicle is far from the CCP, it returns a negligible value. Considering states of all vehicles, they solved a constrained nonlinear optimization problem under a model predictive control (MPC) framework in order to let the vehicles cross the intersection rapidly by minimizing the total quantified risks of all vehicle pairs. However, MPC is usually computationally demanding. In their experiment, the average computation time is found to be about 1.76 s per iteration, which is obviously not practical in real world application.

To reduce computational overhead, Dai *et al.* first quantitatively analyzed the characteristics of vehicle movement and transformed their factors into a convex objective function. Furthermore, they defined the decision zone and divided an intersection into multiple collision areas. Then they designed a schedule rule to determine the priority of the vehicles in each collision area, which linearizes the collision constraints. On this basis, they transformed the traffic control model into a convex optimization, which can be solved efficiently in real time [12]. However, an additional algorithm, in the form of priority rules for each collision area, is required to deal with the linearization of nonlinear constraints.

Reinforcement learning (RL) has the potential to solve low computation efficiency suffered by current centralized coordination methods. Given a reward function, RL learns a policy by trial-and-error within a simulated environment or the real world to maximize the sum of future rewards. Then the learned policy can be used to get real time decisions compared with solving an optimization problem in each step. Existing RL research on autonomous driving mostly focus on the intelligence of single-vehicle driving in relatively simple traffic scenarios [13]–[19]. In this paper, we employ RL as our method for centralized coordination of multiple connected vehicles to realize autonomous passing at intersections without traffic signals. We first propose an RL training algorithm, model accelerated proximal policy optimization (MA-PPO), which incorporates a prior model into proximal policy optimization (PPO) algorithm to enhance sample efficiency. Then we present the design of state, action and rewards to formulate centralized coordination as an RL problem, where the rewards include a collision punishment and a step punishment for safety and efficiency considerations. Finally, a simulation-based case study implemented on a four-way single-lane approach intersection showed that our method spends only 1/400 of the computing time of VICS and increases the efficiency of the intersection by 4.5 times.

The main contributions of this paper are summarized as follows. First, we use reinforcement learning in centralized coordination of connected vehicles at an intersection without traffic signals and reduce computation overhead significantly. Second, we propose model accelerated proximal policy optimization algorithm, which incorporates a prior model into proximal policy optimization to accelerate the learning process in terms of sample efficiency. Third, we conduct extensive simulations, and the results demonstrate the superiority of our proposed method.

The rest of this paper is organized as follows. Section II introduces the preliminaries of Markov decision process (MDP) and policy gradient methods. Section III proposes MA-PPO, which is an improvement based on PPO and model-based RL methods. Section IV illustrates our problem statement and methodology and section V looks into experimental settings and illustrates results. Last section VI summaries this work.

II. Preliminaries of RL

Consider an infinite-horizon discounted MDP, defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, d^0, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are state and action spaces respectively. We suppose that $\mathcal{S}$ and $\mathcal{A}$ are compact subsets of $\mathbb{R}^n$ and $\mathbb{R}^m$. $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the transition probability distribution. $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, $d^0 : \mathcal{S} \to \mathbb{R}$ is the distribution of the initial state s_0, and $\gamma \in (0, 1)$ is the discount factor.

Let π denote a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, we seek to learn the optimal policy π^* which has maximum values $v^{\pi^*}(s)$ for all $s \in \mathcal{S}$, where the value function $v^\pi(s)$ is the expected sum of discounted rewards from a state when following policy π:

$$v^\pi(s) = \mathbb{E}_{a_t, s_{t+1}, \ldots} \left\{ \sum_{l=t}^{\infty} \gamma^{l-t} r_l | s_t = s \right\},$$

where $a_t \sim \pi(a_t|s_t), s_{t+1} \sim p(s_{t+1}|s_t, a_t)$ and $r_t := r(s_t, a_t, s_{t+1})$ for short. Similarly, we use the following standard definition of the state-action value function $q^\pi(s, a)$:

$$q^\pi(s, a) = \mathbb{E}_{s_{t+1}, a_{t+1}, \ldots} \left\{ \sum_{l=t}^{\infty} \gamma^{l-t} r_l | s_t = s, a_t = a \right\}.$$

A. Vanilla Policy Gradient

In practice, finding the optimal actions for all states is impractical for problems with large state space. We instead use a parameterized policy $\pi_\theta(a|s)$ and try to seek the optimal parameter vector θ. For the same reason, state-value function is parameterizd as $V(s, w)$ with a parameter vector w. Policy optimization methods seek to find optimal θ^* which maximize average performance of policy π_θ, i.e.,

$$J(\theta) = \mathbb{E}_{s_0, a_0, \ldots} \left\{ \sum_{t=0}^{\infty} \gamma^t r_t \right\}, \tag{1}$$

where $s_0 \sim d^0(s_0), a_t \sim \pi_\theta(a_t|s_t), s_{t+1} \sim p(s_{t+1}|s_t, a_t)$ and $r_t := r(s_t, a_t, s_{t+1})$.

Vanilla methods optimize (1) by stochastic policy gradient [20]. Its gradient is shown as

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim d^\gamma, a \sim \pi_\theta} \left\{ \nabla_\theta \log \pi_\theta(a|s) q^{\pi_\theta}(s, a) \right\}, \tag{2}$$

where d^γ is called discounted visiting frequency, which in practice is usually replaced with the stationary state distribution

under π_θ denoted by d^{π_θ} [21]. Combined with the baseline technique [22], we can write (2) in format

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{s\sim d^{\pi_\theta}, a\sim\pi_\theta}\left\{\nabla_\theta \log \pi_\theta(a|s) A^{\pi_\theta}(s,a)\right\},$$

where $A^{\pi_\theta}(s,a) := q^{\pi_\theta}(s,a) - v^{\pi_\theta}(s)$ is the advantage function which could be estimated by several methods [23].

B. Trust Region Method

While the vanilla policy gradient is simple to implement, it often leads to destructively large policy updates. Trust region policy optimization (TRPO) optimizes a lower bound of (1) to guarantee performance improvement, i.e.,

$$\max_\theta \mathbb{E}_{s,a}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A^{\pi_{\theta_{\text{old}}}} - \beta \text{KL}\left[\pi_{\theta_{\text{old}}}(\cdot|s), \pi_\theta(\cdot|s)\right]\right]. \quad (3)$$

However, it is difficult to choose a single value of β that performs well across different problems, TRPO uses a constraint instead, shown as

$$\begin{aligned} \max_\theta \quad & \mathbb{E}_{s,a}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A^{\pi_{\theta_{\text{old}}}}\right] \\ \text{s.t.} \quad & \mathbb{E}_s\left[D_{\text{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s) \| \pi_\theta(\cdot|s)\right)\right] \le \delta, \end{aligned}$$

where δ is a bound on the KL divergence between the new policy and the old policy, $\mathbb{E}_{s,a}[\ldots] := \mathbb{E}_{s\sim d^{\pi_{\theta_{\text{old}}}}, a\sim\pi_{\theta_{\text{old}}}}[\ldots]$, $\mathbb{E}_s[\ldots] := \mathbb{E}_{s\sim d^{\pi_{\theta_{\text{old}}}}}[\ldots]$ and $A^{\pi_{\theta_{\text{old}}}} := A^{\pi_{\theta_{\text{old}}}}(s,a)$. To solve the constrained optimization problem, TRPO linearizes the objective function and transforms the constraint into a quadratic constraint. Because it is time-consuming to solve the inverse of Hessian matrix for large-scale problems, in each iteration, the authors use the conjugate gradient method to solve a system of linear equations to obtain the feasible ascent direction, and then obtain the optimal solution through the line search method. TRPO can be regarded as natural policy gradient methods [24]. It finds the steepest policy gradient in the fisher matrix normed space rather than the euclidean space, which helps to reduce impact of policy parameterization and stabilize the learning process. However, the conjugate gradient method leads to a fixed number of inner iterations every time the feasible direction is solved, which increases the complexity of the algorithm.

III. Model Accelerated PPO

A. Proximal Policy Optimization

In this paper, the PPO algorithm is employed as our baseline. It is inspired by TRPO and has two main differences, namely, unconstrained surrogate objective function and generalized advantage estimation.

1) Unconstrained Surrogate Objective Function: Observing monotonic policy improvement requires punishment of policy deviation from the theroy of TRPO, PPO alternatively constructs an unconstrained surrogate objective function to remove the incentive for large policy updates. Its objective is shown as

$$J^{\text{ppo}}(\theta) = \mathbb{E}_{s,a}\left[\min\left(\rho_\theta A^{\pi_{\theta_{\text{old}}}}, \text{clip}\left(\rho_\theta, 1-\epsilon, 1+\epsilon\right) A^{\pi_{\theta_{\text{old}}}}\right)\right], \quad (4)$$

where $\rho_\theta = \frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}$. When $A^{\pi_{\theta_{\text{old}}}} > 0$, the term ρ_θ would tend to be much larger than 1 to make the performance as high as possible, which leads to unstable learning. The objective of PPO (4) cuts this motivation by clipping ρ within $1+\epsilon$. Same situation is with $A^{\pi_{\theta_{\text{old}}}} < 0$.

2) Generalized Advantage Estimation: The advantage function is necessary for the calculation of the PPO policy gradient, and it can be estimated by

$$\hat{A}^\pi(s,a) = \hat{Q}^\pi(s,a) - V(s,w),$$

where $\hat{Q}^\pi(s,a)$ is the action-value function estimated by samples, $V(s,w)$ is the approximation of the state-value function. TRPO uses Monte Carlo methods to construct $\hat{Q}^\pi(s,a)$, i.e.,

$$\hat{Q}^\pi(s_t,a_t) = \sum_{l=t}^{\infty} \gamma^{l-t} r_l.$$

It is unbiased but suffers from high variance. Actor-critic methods use one-step bootstrapping to form $\hat{Q}^\pi(s,a)$, i.e.,

$$\hat{Q}^\pi(s_t,a_t) = r_t + \gamma V(s_{t+1},w),$$

which is biased but has low variance.

Generalized advantage estimation uses the linear combination of n-step bootstrapping to obtain both low bias and low variance, which is shown as

$$\hat{Q}^\pi(s_t,a_t) = \sum_{l=t}^{\infty} (\gamma\lambda)^{l-t}\delta_l + V(s_t,w),$$

where δ_l is the TD error,

$$\delta_l = r_l + \gamma V(s_{l+1},w) - V(s_l,w).$$

Compared with TRPO, PPO is much simpler and faster to implement because it is a first-order optimization algorithm and has better convergence speed when it is combined with generalized advantage estimation. However, PPO is an on-policy method and inevitably has high sample complexity.

B. Model-Based RL

To reduce sample complexity, recent RL algorithms have been proposed to incorporate a given or learned environment model as an additional source of information, where the environment models are models of the true transition dynamics p. Generally, there are two ways to use environment models: value gradient methods, or using models for imagination.

Value gradient methods link together the policy, model, and reward function to compute an analytic policy gradient by back-propagation of reward along a trajectory [25]–[27]. A major limitation of this approach is that the dynamic model can only be used to retrieve information already presented in observed data, and albeit with lower variance, the actual improvement in efficiency is relatively small. Alternatively, the given or learned model can also be used for imagination. This usage can be naturally incorporated in the model-free RL framework. However, the inconsistency between models and true dynamics can lead to large errors when the model-generated rollouts are too long [28], [29].

C. Model Accelerated PPO

PPO is a model-free on-policy RL algorithm. Model-free means it knows nothing about the environment and can only learn from interactions with the environment. As a result, it inevitably requires a large amount of experience data to learn the value function and policy from scratch. Even worse, the on-policy property makes experiences produced by previously trained policies useless, which aggravates sample inefficiency. The large demand for on-policy samples will lead to a large amount of time to interact with the environment, as well as the safety issues of training in the real world. This is our motivation to accelerate PPO in terms of sample efficiency.

In order to alleviate the low sample efficiency suffered by PPO, we employ the environment model in the training process, and propose model accelerated PPO (MA-PPO), which can not only learn from the interaction with the environment, but also learn from the model. In MA-PPO, the environment model is used for imagination, that is, we use the model to generate virtual samples, and then use the algorithm to learn from these samples. In this way, the objective function of PPO can be used without any change, and the environment model and PPO algorithm can be naturally combined. By learning from the data generated by the environment model, MA-PPO can reduce the dependence on real samples generated by interactions with a simulator or the real world. When the model error is small, MA-PPO can achieve better performance using fewer real samples, enhancing sample efficiency of PPO.

A big concern is that although learning from the data generated by the model can greatly improve the sample efficiency, the final performance of the algorithm will be affected by the existence of model errors. Considering this problem, MA-PPO controls for uncertainty in the model by only allowing imaging to a fixed depth, because short rollouts are more likely to reflect the real dynamics, reducing the opportunities for policies to rely on inaccuracies of model predictions. The depth is set to be a hyperparameter and can be determined according to the error of the model in problems.

MA-PPO algorithm is shown in algorithm 1. The first line of the algorithm is used to initialize network parameters and algorithm parameters. The second line starts the main loop of the algorithm, which is divided into two main parts. The first part describes the algorithm learning from samples generated by interacting with the environment (lines 3-16). The emphasis is on estimating the advantage function and the value target (lines 7-8) using generalized advantage estimation method, and then they are used to calculate the gradient of the policy and the value function respectively (lines 11-16). The second part describes the algorithm learning through virtual samples generated by the environment model (lines 17-28). First, a batch of data is extracted from the buffer (line 17), and then the model and policy are used to extend each data point in the batch to the depth D, where the depth is used to control uncertainties in the model. The estimated advantage functions and value function targets are calculated through these virtual samples (lines 18-23). Finally, the policy gradient and value function gradient are calculated, and the corresponding parameters are updated (lines 25-28).

Algorithm 1: MA-PPO

1 Randomly initialize value network $V(s, w)$ and policy network π_θ with weights w and θ, set $\lambda, \gamma, \epsilon$, total timesteps T_{total}, batch size B, minibatch size MB, epoch U, model rollout depth D, buffer $\mathcal{B} = \varnothing$
2 **for** *iteration = 1, 2, ⋯, T_{total}/B* **do**
3 Run several episodes using policy π_θ to collect B timesteps $\mathcal{D} = (s_i, a_i, r_i, s_{i+1})_{i=0:B-1}$
4 $\mathcal{B} = \mathcal{B} \cup \mathcal{D}$
5 Calculate TD errors δ_i, $i = 0, 1, \cdots, B-1$
6 **for** *i=0, 1, ⋯, B-1* **do**
7 $\hat{A}_i = \sum_{l \geq i}^{End\_episode} (\lambda\gamma)^{l-i} \delta_l$
8 Estimate value target $\hat{V}_i = \hat{A}_i + V(s_i, w)$
9 **end**
10 $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$
11 **for** *epoch = 1, 2, ⋯, U* **do**
12 $J^{\text{PPO}}(\theta) = \frac{1}{B} \sum_{i=1}^{B} \min(\frac{\pi_\theta(a_i|s_i)}{\pi_{\theta_{\text{old}}}(a_i|s_i)} \hat{A}_i, \text{clip}(\frac{\pi_\theta(a_i|s_i)}{\pi_{\theta_{\text{old}}}(a_i|s_i)}, 1-\epsilon, 1+\epsilon)\hat{A}_i)$
13 Update θ by $\nabla_\theta J^{\text{PPO}}$
14 $J^{\text{Value}}(w) = -\frac{1}{B} \sum_{i=1}^{B} (\hat{V}_i - V(s_i, w))^2$
15 Update w by $\nabla_w J^{\text{Value}}$
16 **end**
17 Sample $\mathcal{D} = (s_i, a_i, r_i, s_{i+1})_{i=0:B-1}$ from $\mathcal{B}$
18 **for** *i=0, 1, ⋯, B-1* **do**
19 Extend $\hat{s}_0 = s_i$ by the policy π_θ and the environment model to the depth D. Denote virtual samples as $(\hat{s}_t, \hat{a}_t, \hat{r}_t, \hat{s}_{t+1})_{t=0:D-1}$
20 Calculate TD errors $\hat{\delta}_t, t = 0, 1, \cdots, D-1$ by virtual samples
21 $\hat{A}_i = \sum_{l=0}^{D} (\lambda\gamma)^l \hat{\delta}_l$
22 Estimate value target $\hat{V}_i = \hat{A}_i + V(s_i, w)$
23 **end**
24 $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$
25 **for** *epoch = 1, 2, ⋯, U* **do**
26 Update θ by $\nabla_\theta J^{\text{PPO}}$
27 Update w by $\nabla_w J^{\text{Value}}$
28 **end**
29 **end**

IV. Problem Statement and Formulation

A. Problem Statement

The scenario is a typical four-way single-lane intersection without traffic signals, as shown in Fig. 1. Each direction is denoted by its location in the figure, i.e. up (U), down (D), left (L) and right (R) respectively. We only focus on vehicles within a certain distance from the intersection center. Vehicles in each entrance are allowed to turn right, go straight or turn left. There are 12 types of vehicles, denoted by their entrances and exits, i.e. DR, DU, DL, RU, RL, RD, LD, LR, LU, UL, UD, and UR. Their number and meanings are listed in Table I . All their possible conflict relations are also illustrated in Fig. 1, which can be categorized into three classes, including crossing conflicts

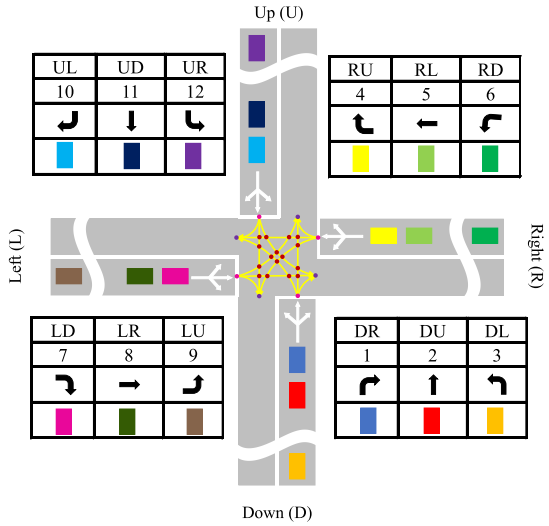


Fig. 1. Intersection scenario settings.

TABLE I
DIFFERENT TYPES OF VEHICLES

Type	Number	Meaning
DR	1	From 'Down' turn right to 'Right'
DU	2	From 'Down' go straight to 'Up'
DL	3	From 'Down' turn left to 'Left'
RU	4	From 'Right' turn right to 'Up'
RL	5	From 'Right' go straight to 'Left'
RD	6	From 'Right' turn left to 'Down'
LD	7	From 'Left' turn right to 'Down'
LR	8	From 'Left' go straight to 'Right'
LU	9	From 'Left' go straight to 'Up'
UL	10	From 'Up' turn right to 'Left'
UD	11	From 'Up' go straight to 'Down'
UR	12	From 'Up' turn left to 'Right'

TABLE II
REWARD SETTINGS

Reward items	Reward
Collision	-50
Step reward	-1
Some vehicle passes	10
All vehicles pass	50

(denoted by red dots), converging conflicts (denoted by purple dots), and diverging conflicts (denoted by pink dots). To simplify our problem, we choose eight modes out of all the twelve modes to cover main conflicts. The rules are set as follows. First, each entrance includes two vehicles, in which one of them is a right-turning vehicle and the other is a vehicle turning left or going straight. Right-turning vehicles will only lead to converging and diverting conflicts, while vehicles going straight or turning left will introduce crossing conflicts. In the four entrances, two of them (entrances D and L) include a left-turning vehicle, and the other two (entrances U and R) include a vehicle going straight, which leads to a large number of crossing conflicts. In addition, the location and order of different vehicles in each entrance are random, which makes the problem still have enough complexity.

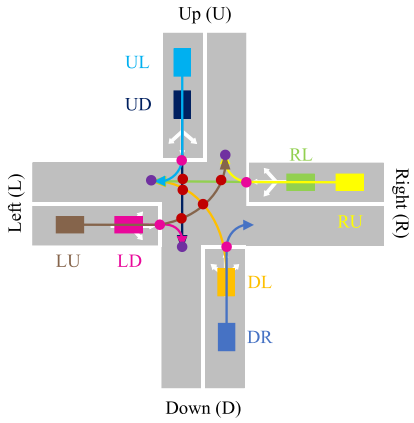


Fig. 2. Vehicle modes chosen for the experiments.

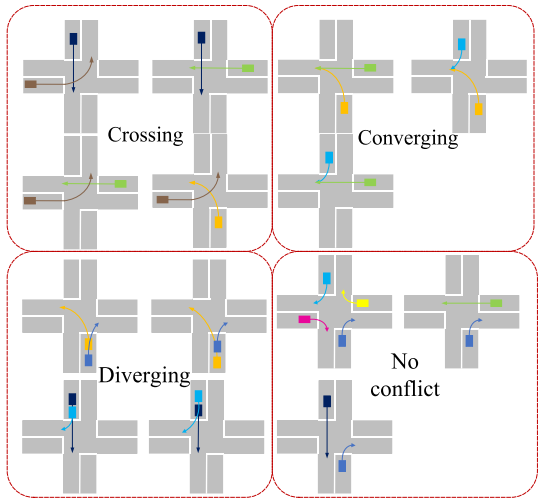


Fig. 3. Typical modes of conflicts in the experiments.

The eight modes include DR, DL, RU, RL, LD, LU, UL, UD, as shown in Fig. 2. From the figure, we can summary all types of conflicts it contains, which is shown in Fig. 3.

We adopt the following assumptions. First, all vehicles are equipped with positioning and velocity devices so that we can gather location and movement information when they enter the interesting zone of the intersection. Second, only the longitudinal motion of the vehicles is controlled, assuming the vehicles follow the lateral trajectories. The environment model used in our MPC baseline and MA-PPO describing the longitudinal motion of vehicle i is given by

$$x_i^{\text{long}}(t+1) = x_i^{\text{long}}(t) - v_i(t)\tau - \frac{1}{2}a_i(t)\tau^2$$
$$v_i(t+1) = v_i(t) + a_i(t)\tau, \tag{5}$$

where x_i^{long} is the longitudinal distance from the intersection, v_i and a_i are the velocity and acceleration, τ is the discrete time step. For the true dynamics of the simulation, the longitudinal motion x_i^{long} is added with a Gaussian noise $\psi_i \sim \mathcal{N}(0, \sigma_i^2)$. The standard deviation σ_i is related to the velocity, i.e., $\sigma_i = c_1 v_i \tau + c_2$, where c_1 controls the magnitude of the noise and c_2 is a small number to prevent numerical instability. The problem

is stated as follows: In each step, given positions and velocities of all the vehicles from the simulation, the acceleration of each vehicle needs to be decided in real time to let all vehicles pass the intersection safely and quickly.

B. MPC and RL

In this paper, we use VICS as a baseline for our comparison, which is a coordination scheme in the MPC framework. The scheme efficiently utilizes the intersection area by preventing each pair of conflicting vehicles from approaching their cross-collision point (CCP) at the same time, where the CCP is the intersection of their trajectories. A risk function is proposed to quantifying the risk of a collision of a pair of vehicles around their CCP, which is given by

$$R_{i,j}(t) = H\delta_{i,j}e^{-(\alpha_i d_i^2+\alpha_j d_j^2)}. \tag{6}$$

In this function, H is a positive constant indicating the highest possible risk of collision, and $\delta_{i,j}$ is a binary variable to state whether the vehicles i and j have a CCP. Besides, d_i and d_j are the distances of the CCP from current position of vehicles i and j along their trajectories, and α_i and α_j are positive constants. At any time, if two conflicting vehicles are very close to their CCP, the risk function returns a high value, and if at least one vehicle is far from the CCP, it returns a low value. Based on that, a constrained nonlinear optimization problem is constructed as follows,

$$\begin{aligned} J = &\sum_{t=0}^{T-1}\sum_{i=1}^{N} w_v(v_i(t+1)-v_d)^2 \\ &+ \sum_{t=0}^{T-1}\sum_{i=1}^{N} w_a(a_i(t))^2 + \sum_{t=0}^{T-1}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\mathcal{R}_{i,j}(t) \\ &\text{s.t. } v_{\min} \le v_i \le v_{\max}, \\ &\quad a_{\min} \le a_i \le a_{\max} \end{aligned} \tag{7}$$

where T is the length of the prediction horizon, v_d is the desired velocity, and w_v and w_a are weight coefficients. There are three cost terms in total. The first term denotes the cost related to velocity deviation from the desired value v_d. The second term denotes the cost of acceleration. Minimizing these two terms means comfortable and smooth flow of vehicles. The third term denotes the cost related to the risk of collisions as defined in the risk function (6), which sums up quantified risks at all CCPs for all possible pairs of vehicles considering their predicted trajectories in the horizon. Besides, constraints that are related to the velocity and acceleration limits are defined.

The constrained nonlinear optimization problem in the proposed MPC framework is solved in the SciPy package in python by the optimization toolbox *minimize* using the Sequential Least Squares Programming (SLSQP). The prediction horizon is chosen as 20 steps. Detailed parameter settings are listed in Table III.

Reinforcement learning comes from dynamic programming methods of optimal control, which is used to solve the optimal sequential decisions. Similar to MPC, which uses the model to minimize the sum of finite time-domain losses to optimize future sequential controls, reinforcement learning optimizes a policy to maximize the sum of future rewards. The difference is that MPC methods need to solve optimization problems online, while reinforcement learning can solve a policy offline and apply it online.

TABLE III
HYPERPARAMETERS OF THE EXPERIMENTS

Parameters	Value
Simulator	
Discrete time step τ	0.1s
Vehicle numbers N	8
Simulation noise	c_1=1/30, c_2=2e-7
MA-PPO & PPO	
Discount factor γ	0.99
λ	0.95
Clip range ϵ	0.2
Total timesteps T_{total}	5e7
Seed number	5
Rollout depth D *	50
Batch size B	2048
Minibatch size MB	64
Epoch U	10
Learning rate LR	$0.0003 \rightarrow 0$
Hidden layer number	2
Hidden units number	128
Optimizer	Adam
Number of workers	16
VICS	
Predictive horizon T	20
Maximum velocity $v_{\max}$	$10\mathrm{m/s}$
Minimum velocity $v_{\min}$	$0\mathrm{m/s}$
Maximum acceleration $a_{\max}$	$5\mathrm{m/s}^2$
Minimum acceleration $a_{\min}$	$-5\mathrm{m/s}^2$
Desired velocity v_d	$8\mathrm{m/s}$
w_v	1
w_a	5
H	1000
α	0.005

*Parameter used only by MA-PPO.

C. RL Formulation

We are ready to transform our problem to an RL problem by defining state space, action space and reward function, which are basic elements in RL.

1) State and Action Space: By our assumption, we need to control at most eight vehicles at a time, i.e. two different types of vehicles at each entrance. Both the state and the action are defined as the concatenation of the state and action of each vehicle, i.e.,

$$s = [s_{\mathrm{DR}}, s_{\mathrm{DL}}, s_{\mathrm{RU}}, s_{\mathrm{RL}}, s_{\mathrm{LD}}, s_{\mathrm{LU}}, s_{\mathrm{UL}}, s_{\mathrm{UD}}]$$
$$a = [a_{\mathrm{DR}}, a_{\mathrm{DL}}, a_{\mathrm{RU}}, a_{\mathrm{RL}}, a_{\mathrm{LD}}, a_{\mathrm{LU}}, a_{\mathrm{UL}}, a_{\mathrm{UD}}],$$

where s_* and a_* denote the state and action of the vehicle type *.

The state of each vehicle should contain position and velocity information. Intuitively, we can form the state by a tuple

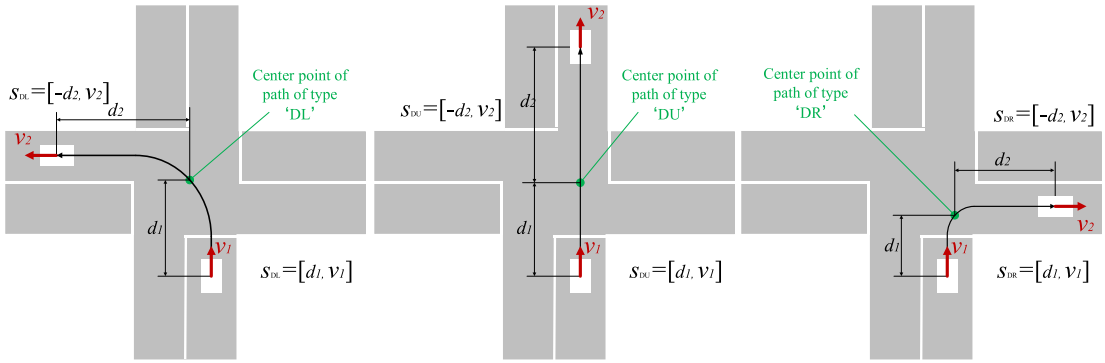


Fig. 4. State formulation.

of coordinate and velocity, i.e. (x, y, v), where (x, y) is the coordinate of its position and v is the velocity. However, by our task formulation, each vehicle has a fixed lateral trajectory corresponding to its type. There will be redundant information if we use this state formula. Besides, for continuous states, it is necessary to decrease the state space dimensionality to speed up learning and enhance stability. Observing all the trajectories are cross the intersection, we further compress the state of each vehicle by (d, v), where d is the distance between a vehicle and the center of its trajectory. Note that d is positive when the vehicle is heading for the center and negative when it is leaving. The state formulation is shown as Fig. 4.

For the action, we use the acceleration of each vehicle. Finally, a 16-dimensional state space and an 8-dimensional action space are constructed.

2) Reward Settings: Reward functions design concerns with convergence speed and asymptotic performance of RL algorithms. There are two points to note when defining reward functions. First, the complexity of the reward function and the convergence speed should be balanced. The more detailed the definition of the reward function, the faster the algorithm will learn, but defining such a reward function will require a lot of human design, especially in high-dimensional state space and action space. Second, positive and negative rewards have different functionality. Positive rewards drive the system to avoid terminals to maximize accumulating rewards unless the terminal yields a large positive reward, while negative rewards incentive the agent to reach a terminal state as quickly as possible to avoid accumulating penalties.

For our high-dimensional RL problem, a look-up table is adopted as the reward function, which is much easier to design compared with a continuous function. The reward function is designed under consideration of safety, efficiency and task completion. First of all, the task is designed in an episodic manner, in which a bad terminal state and a good terminal state are given, i.e., collision and all vehicles passing the intersection. In order to learn a safe policy, when the system reaches the bad terminal state (collision), we set up a large negative reward to discourage its presence. In order to improve traffic efficiency, we set a small negative reward at each step to drive the system to the good terminal state quickly. To encourage task completion, we set a positive reward as long as some vehicle passes the intersection and a large positive reward when the good terminal state is reached (all vehicles passing the intersection). All reward settings are listed in Table III.

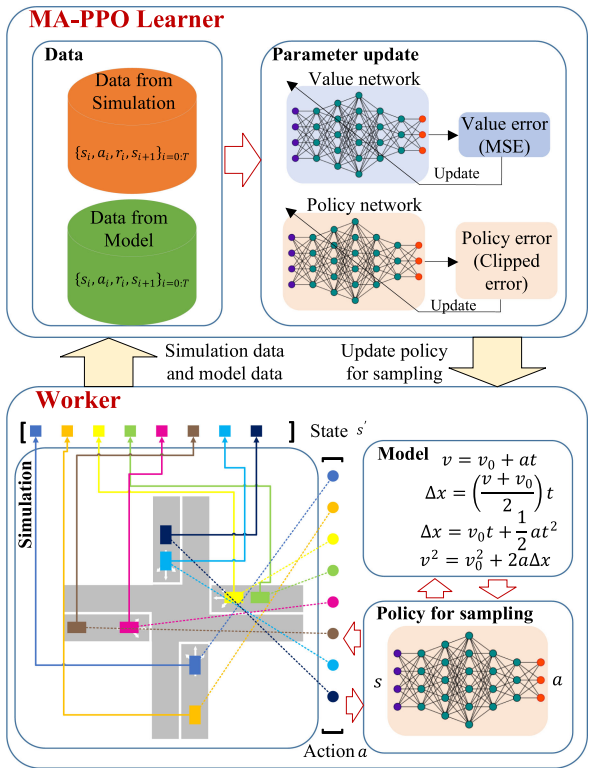


Fig. 5. Overall architecture of MA-PPO.

D. Algorithm Architecture

In this section, we illustrate how to apply MA-PPO algorithm to this centralized coordination problem. The MA-PPO architecture consists of two main parts, including MA-PPO learner and worker. The worker is in charge of getting the updated policy from the learner and using it to collect experience data from simulation and the environment model. MA-PPO learner then uses the data collected from the worker to update the value and policy networks by gradient descent methods, and finally syncs the updated parameters to the worker for the next iteration. The overall architecture is shown in Fig. 5.

V. Experiments

A. Experimental Settings

Four sets of experiments are conducted in this section. In the first experiment, we train a policy by MA-PPO and compare it with a policy from the early training stage to show improvements over the training process. The second experiment compares computing time and traffic efficiency of the policy trained by MA-PPO with that of VICS. The third experiment illustrates the training process of MA-PPO and PPO, which shows our algorithm has better sample efficiency. The fourth experiment shows the performance comparison between MA-PPO and PPO under different environment noise, and shows that we can control the performance loss caused by the model inaccuracy through adjusting the depth of model rollouts in MA-PPO. All the experiments are carried out in a simulation environment, in which the scenario is shown in Fig. 2. The initial positions of all vehicles are random. Considering it is a small intersection in our experiments, we only verify our algorithm in a low-speed scene. The maximum speed of all vehicles is limited to 10 m/s.

Detailed parameter settings of all these experiments are listed in Table III. The time step of all simulation experiments is set to be 0.1 s, and the number of vehicles is set to be 8. Except for the fourth experiment, the environment noise related parameters c_1 and c_2 are set to be 1/30 and 2e-7 respectively, which are defined in the Section IV. We employ multiple layers percepton with two hidden layers as the approximate functions of the policy and the value function. Both of them have 128 units in each hidden layer, And the policy network has 16 output units to parameterize the Gaussian distribution of the accelerations of all vehicles, while the value function has only one output unit for the state value. In each iteration, each worker collects $B = 2048$ transitions and uses minibatch size $MB = 64$ and epoch $U = 10$ for updating. To stabilize the training process, the learning rate LR is set to linearly decrease from 0.0003 to 0, and Adam is selected as the optimizer to stabilize the gradient direction. The training process is not terminated until the number of collected real samples reaches 5e7. Sixteen parallel workers are used to improve exploration and stabilize the learning process. In each iteration, each worker collects a batch of data, then takes the first minibatch to calculate the gradient, then the global gradient is obtained by averaging all local gradients of workers. Each worker updates its parameters using the global gradient and goes on like this. Rollout depth D is only a parameter of MA-PPO, which is used to control the depth of model rollouts. Except for the fourth experiment, we set it to be 50. The second experiment involves the comparison of MA-PPO and VICS. The parameters of VICS refer to the settings in its original paper. The prediction horizon is set to be 20 steps, and the vehicle speed is limited to the interval [0, 10 m/s], and the acceleration is limited to the interval [-5$\mathrm{m/s^2}$, 5$\mathrm{m/s^2}$]. The desired velocity v_d is set to 8 m/s. Besides, the weights of the velocity term and the acceleration term in the objective function is $w_v = 1$ and $w_a = 5$ respectively, and the parameters of the risk term are $H = 1000$, $\alpha = 0.005$.

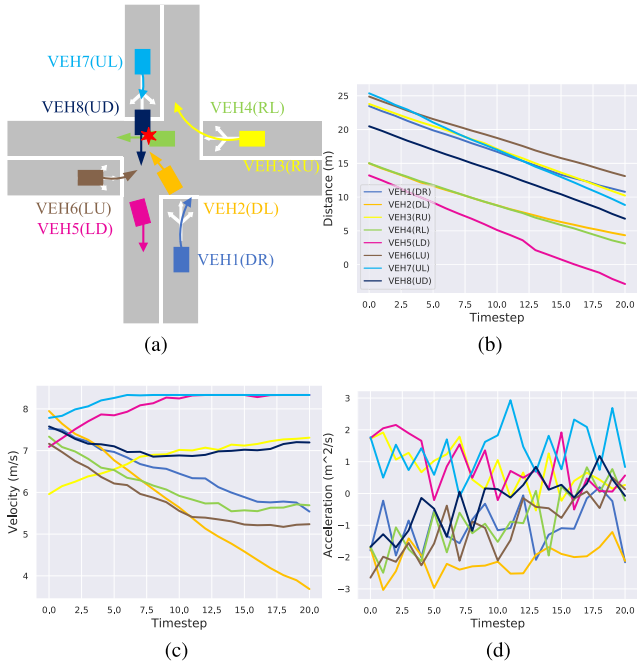


Fig. 6. Results from the 20th iteration of the training process, including the distance to the intersection center, the speed information and the action they take during the episode. (a) An episode with collision. (b) Distance vs Timestep. (c) Velocity vs Timestep. (d) Action vs Timestep.

B. Comparison of Results in Different Training Stages

In this experiment, we train a policy by MA-PPO and then compare it with a policy from the early training stage.

Fig. 6 visualizes an episode in the 20th iteration of the training process. In this episode, VEH5 (mode: LD) passes the intersection successfully. However, VEH4 (mode: RL) and VEH8 (mode: UD) collide. From Fig. 6(b) we can see all the eight vehicles are approaching the center of the intersection, but almost none of them realize to decelerate to avoid collision except for VEH2. During the last few steps before the collision, the velocities of VEH4 and VEH8 still maintain their trend without significant changes. The random curves in the acceleration illustrated in Fig 6(d) also show that the learned policy at this point fails to coordinate all the vehicles.

Fig. 7 illustrates the result after 500 iterations of training, which shows a good coordination scheme has been learned. In this episode, VEH3 (mode: RU) first passes the intersection. VEH8 slows down before the timestep 26 to wait for VEH7 to turning right. Besides, VEH2 (mode: DL) keeps a low speed to wait for the pass of VEH7. Also, VEH4 (mode: RL) decelerates to wait for VEH2 turning left first. One reasonable explanation that VEH8 has to wait and pass lastly is that it has a longer distance to the center of the intersection than any other vehicles, as shown in Fig. 7(b). It can be seen from Fig. 7(d) that, VEH8 remains stable deceleration between -2$\mathrm{m/s^2}$ and -1$\mathrm{m/s^2}$ until the timestep 26, when VEH7, VEH2 and VEH4 have passed the central area of the intersection.

VEH5 and VEH6 have similar velocity curves. In the beginning, they slow down and keep low velocity until VEH2, VEH3

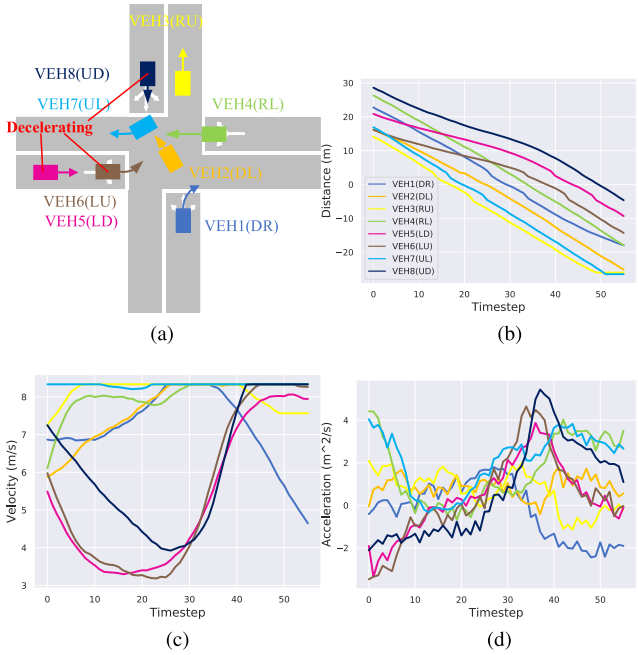


Fig. 7. Results from the 500th iteration of the training process, including the distance to the intersection center, the speed information and the action they take during the episode. (a) An episode without collision. (b) Distance vs Timestep. (c) Velocity vs Timestep. (d) Action vs Timestep.

and VEH4 pass the intersection. After the timestep 25, both of them begin to speed up and pass the intersection because there is no risk of collision around the center of the intersection. On the other hand, the velocity curves of VEH2, VEH3 and VEH4 demonstrate that they tend to keep a high speed so that they could pass the intersection quickly. To sum up, Fig. 7 shows that MA-PPO has learned a human-like policy in the 500th iteration, which can avoid the potential collision, and coordinate all the vehicles passing the intersection efficiently.

C. Comparison of MA-PPO Policy and VICS

In this experiment, we implement VICS in our experiment and compare traffic efficiency and computing time with the policy trained by MA-PPO.

For the implementation of VICS, we follow the method in the original paper. First, we define the CCP for each pair of conflicting vehicles in our scenario. Then we formulate the nonlinear optimization problem (7) with coefficients shown in Table III. All evaluations were performed on a single computer with a 2.9 GHz Intel(R) Core(TM) i9-8950HK CPU.

We use the average episode length and the average computing time as metrics of traffic and computing efficiency respectively. The average episode length is the average number of timesteps over 10 episodes, where an episode means the process from initialization to all vehicles passing the intersection. The average computing time is the average calculation time spent in all timesteps over 10 episodes. Besides, we also compute the variance of the computing time. It is obvious to see that the shorter the average episode length, the higher the traffic efficiency, and the less the average computing time, the lower the overhead.

TABLE IV
COMPARISON OF OUR METHOD AND VICS

	Ours	VICS
Average episode length	53.44	241.0
Average computing time	0.00417s	1.64560s
Variance of computing time	0.000247	13.657

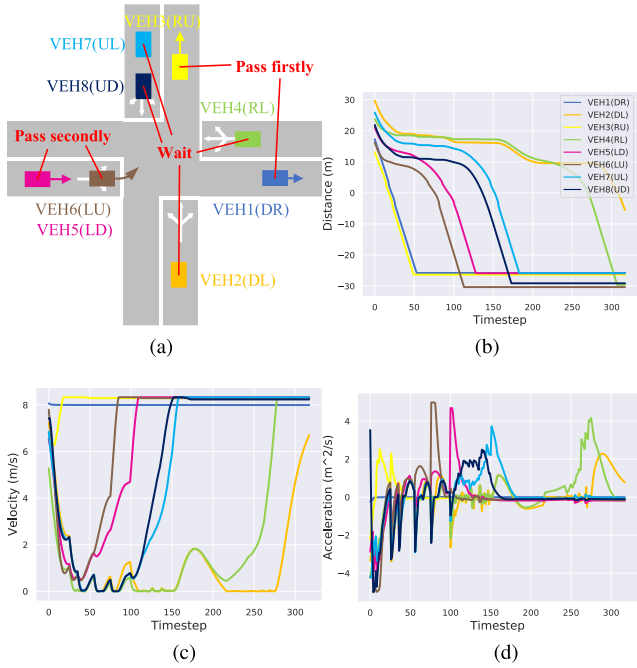
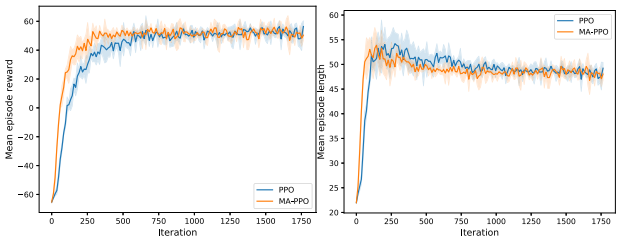


Fig. 8. A typical episode of VICS, including the distance to the intersection center, the speed information and the action taken during the episode. (a) An episode using VICS. (b) Distance vs Timestep. (c) Velocity vs Timestep. (d) Action vs Timestep.

The results are shown in Table IV. From the results, it can be seen that the traffic efficiency of our method is 4.5 times that of VICS and our method is around 400 times more efficient than VICS in terms of computing efficiency. Besides, the variance of computing time of VICS is much higher than that of our method, which means the computing time of VICS varies in different timesteps. Actually, the maximum value is up to 30 s in our experiments, which makes it not practical to be applied in the real world. On the other hand, our method can meet the real time requirements while it has better performance.

We also show a typical episode of VICS in Fig. 8. Comparing it with Fig. 7, we can see that VICS tends to stop all the vehicles which have conflict relationship with the vehicles in the intersection. It lets VEH1 (mode: DR) and VEH3 (mode: RU) cross the intersection first, then VEH5 (mode: LD) and VEH6 (mode: LU) accelerate to pass, followed by VEH7 (mode: UL) and VEH8 (mode: UD), and finally VEH2 (mode: DL) and VEH4 (mode: RL) pass the intersection. On the other hand, our method only slows down conflict vehicles to a lower speed rather than makes it pull up. As a result, VICS is less efficient than our approach.



(a) (b)

Fig. 9. Training process of MA-PPO and PPO. (a) Mean episode reward. (b) Mean episode length.

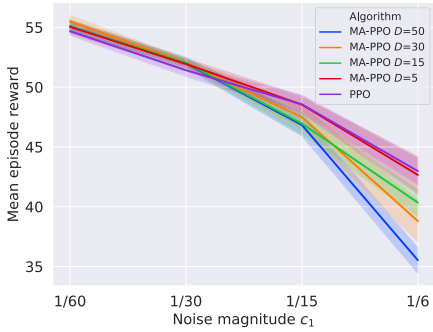


Fig. 10. Asymptotic performance under different environment noise.

D. Sample Efficiency of MA-PPO and PPO

In the third experiment, we use MA-PPO and PPO to train a policy respectively and illustrate their training processes to show the superiority of our proposed algorithm. To eliminate the impact of randomness, we train both PPO and MA-PPO under five random seeds. The results are shown in Fig. 9, where the solid curve represents the mean value and the shaded region represents the variance over different random seeds.

Fig. 9(a) shows the mean episode reward of MA-PPO and PPO during the training process. Both MA-PPO and PPO get the highest reward around 50, which means that all the eight vehicles pass the intersection successfully. Compared with PPO, MA-PPO converges at around 500 iterations, while PPO algorithm needs about 1000 iterations, which shows that MA-PPO converges almost twice as fast as PPO.

Fig. 9(b) shows the change of the mean episode length during the training process. The episode lengths of MA-PPO and PPO first increase rapidly and then reduce to an equal value. This can be explained that at the beginning, the learned policy mainly focuses on how to avoid colliding because this will yield a large negative reward. At that time, the vehicles tend to wait or drive in a very low speed until there is no risk of colliding, which leads to the long episode length. However, such a policy is too conservative and suffers low efficiency. Therefore, the following policy would optimize this process to avoid long waiting time, leading to the decrease of the mean episode length. Similar to Fig. 9(a), MA-PPO obtains faster convergence speed in term of the mean episode length compared with PPO.

E. Asymptotic Performance Under Different Environment Noise

In this experiment, we will explore the influence of different environment noise on the asymptotic performance of PPO and MA-PPO algorithm. As shown in Section IV, the noise in the simulation environment is a Gaussian noise with zero mean value. We adjust the noise by controlling its standard deviation, specifically, c_1. For MA-PPO, the model in (5) is used. With the increase of noise in the environment, the inaccuracy of the model is also increasing. Compared with PPO algorithm, learning with virtual samples will inevitably lead to performance decay. We explore the influence of the hyperparameter D in MA-PPO algorithm on the performance, which is the depth of model rollouts. In the experiment, we take c_1 as 1/60, 1/30, 1/15 and 1/6 to get different environment noise. In addition, the hyperparameter D of MA-PPO algorithm is set to be 50, 30, 15 and 5 respectively. For each group of parameters, we train a policy and run it 15 episodes respectively to obtain the mean episode reward, which is used as the asymptotic performance. The results are shown in Fig. 10.

It can be seen from the figure that for both of PPO and MA-PPO, the asymptotic performance decreases with the increase of noise in the simulation environment, and its variance is also gradually increasing. This is because the uncertainty of the simulation environment increases, as a result, the learned policies tend to be conservative, and the total time of crossing the intersection becomes longer, so the sum of the rewards will also decrease accordingly. In addition, MA-PPO is more robust to small Gaussian errors of the model, for example, when c_1 equals 1/60 or 1/30, no matter which value of D is taken, the performance of MA-PPO is not significantly reduced compared with PPO algorithm. However, when the model error is large (c_1=1/15 or 1/6), the larger the D, the worse the asymptotic performance of MA-PPO. This is because when the model error is large, the uncertainty and inaccuracy of the virtual samples will grow with the increase of rollout depth, which will lead to the rise in the error of the value function and the policy gradient, and finally influence the final performance. Therefore, the hyperparameter D can be adjusted according to the model error in practice to acquire good results on both the sample efficiency and the asymptotic performance.

VI. Conclusion

In this paper, we propose a centralized coordination scheme of automated vehicles at an intersection without traffic signals using reinforcement learning (RL) to address low computation efficiency suffered by current centralized coordination methods, which have been long regarded as a challenging problem due to the real time requirement. We first propose an RL training algorithm, model accelerated proximal policy optimization (MA-PPO), which incorporates a prior model into proximal policy optimization (PPO) algorithm to enhance sample efficiency. Then we present the design of state, action and reward in a typical four-way single-lane intersection which contains eight different modes of vehicles to formulate centralized coordination as an RL problem. Finally, a coordination policy is trained and evaluated through numerical simulation. It is observed that a human-like

policy with high traffic efficiency is got at the end of the training process. We compare its performance with VICS, a coordination scheme based on model predictive control method. Results show that our method spends only 1/400 of the computing time of VICS and increases the efficiency of the intersection by 4.5 times. Besides, we also show that by using a prior model, MA-PPO speeds up the learning process by two times, which shows the superiority of the proposed algorithm.

But our work still needs to be improved. First of all, the trained policy cannot guarantee the safety in real-testing, although there is a strict collision penalty in the reward design and the policy does work well in our experiments. We cannot prove theoretically that our trained policy is collision-free. Second, we only consider the longitudinal dynamics of vehicles while the lateral and longitudinal dynamics are coupled in the real world, which makes the stability of vehicles unable to be guaranteed. In order to solve these problems, we will introduce safety constraints and stability constraints into the original problems in the future work. Since the carrier of our policy is a neural network with a large number of parameters, we will focus on how to solve such a large-scale constrained reinforcement learning problem, such as using penalty function methods or gradient projection methods.

ACKNOWLEDGMENT

We would like to acknowledge Mr. Jingliang Duan and Mr. Zhengyu Liu, for their valuable suggestions throughout this research.

REFERENCES

[1] S. E. Li, S. Xu, X. Huang, B. Cheng, and H. Peng, "Eco-departure of connected vehicles with V2X communication at signalized intersections," *IEEE Trans. Veh. Technol.*, vol. 64, no. 12, pp. 5439–5449, Dec. 2015.

[2] X. Tang, X. Hu, W. Yang, and H. Yu, "Novel torsional vibration modeling and assessment of a power-split hybrid electric vehicle equipped with a dual-mass flywheel," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 1990–2000, Mar. 2017.

[3] T. Liu, X. Tang, H. Wang, H. Yu, and X. Hu, "Adaptive hierarchical energy management design for a plug-in hybrid electric vehicle," *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 11513–11522, Dec. 2019.

[4] Y. Wu, S. E. Li, J. Cortés, and K. Poolla, "Distributed sliding mode control for nonlinear heterogeneous platoon systems with positive definite topologies," *IEEE Trans. Control Syst. Technol.*, vol. 28, no. 4, pp. 1272-1283, Jul. 2020.

[5] F. Gao, X. Hu, S. E. Li, K. Li, and Q. Sun, "Distributed adaptive sliding mode control of vehicular platoon with uncertain interaction topology," *IEEE Trans. Ind. Electron.*, vol. 65, no. 8, pp. 6352–6361, Aug. 2018.

[6] S. E. Li, X. Qin, K. Li, J. Wang, and B. Xie, "Robustness analysis and controller synthesis of homogeneous vehicular platoons with bounded parameter uncertainty," *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 2, pp. 1014–1025, Apr. 2017.

[7] H. Guo, J. Liu, Q. Dai, H. Chen, Y. Wang, and W. Zhao, "A distributed adaptive triple-step nonlinear control for a connected automated vehicle platoon with dynamic uncertainty," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 3861–3871, May 2020.

[8] N. J. Goodall, B. L. Smith, and B. Park, "Traffic signal control with connected vehicles," *Transp. Res. Rec.*, vol. 2381, no. 1, pp. 65–72, 2013.

[9] Y. Feng, K. L. Head, S. Khoshmagham, and M. Zamanipour, "A real-time adaptive signal control in a connected vehicle environment," *Transp. Res. Part C: Emerg. Technol.*, vol. 55, pp. 460–473, 2015.

[10] J. Lee and B. Park, "Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 81–90, Mar. 2012.

[11] M. A. S. Kamal, J.-i. Imura, T. Hayakawa, A. Ohata, and K. Aihara, "A vehicle-intersection coordination scheme for smooth flows of traffic without using traffic lights," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 3, pp. 1136–1147, Jun. 2014.

[12] P. Dai, K. Liu, Q. Zhuge, E. H.-M. Sha, V. C. S. Lee, and S. H. Son, "Quality-of-experience-oriented autonomous intersection control in vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 7, pp. 1956–1967, Jul. 2016.

[13] P. Wolf *et al.*, "Learning how to drive in a real world simulation with deep q-networks," in *Proc. IEEE Intell. Vehicles Symp.*, 2017, pp. 244–250.

[14] Z. Ruiming, L. Chengju, and C. Qijun, "End-to-end control of kart agent with deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, 2018, pp. 1688–1693.

[15] M. Jaritz, R. De Charette, M. Toromanoff, E. Perot, and F. Nashashibi, "End-to-end race driving with deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 2070–2075.

[16] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Proc. Conf. Robot Learn. (CoRL)*, 2017.

[17] D. Jingliang, L. Shengbo, Eben, G. Yang, S. Qi, and C. Bo, "Hierarchical reinforcement learning for self-driving decision-making without reliance on labeled driving data," *IET Intell. Transport Syst.*, vol. 14 no. 5, pp. 297–305, 2019.

[18] A. Kendall *et al.*, "Learning to Drive in a Day," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018.

[19] X. Xiong, J. Wang, F. Zhang, and K. Li, "Combining deep reinforcement learning and safety based control for autonomous driving," 2016, *arXiv:1612.00147*.

[20] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Advances Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.

[21] P. Thomas, "Bias in natural actor-critic algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 441–448.

[22] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[23] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. Int. Conf. Learn. Represent.*, 2016.

[24] S. M. Kakade, "A natural policy gradient," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 1531–1538.

[25] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 465–472.

[26] I. Grondman, "Online model learning algorithms for actor-critic control," Ph.D. dissertation, Delft Univ. Technol., Delft, The Netherlands, 2015.

[27] N. Heess, G. Wayne, D. Silver, T. Lillicrap, T. Erez, and Y. Tassa, "Learning continuous control policies by stochastic value gradients," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2944–2952.

[28] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, "Model-ensemble trust-region policy optimization," in *Int. Conf. Learn. Represent. (ICLR)*, Edinburgh, Oct. 4–5, 2018.

[29] V. Feinberg, A. Wan, I. Stoica, M. I. Jordan, J. E. Gonzalez, and S. Levine, "Model-based value estimation for efficient model-free reinforcement learning," *CoRR*, vol. abs/1803.00101, 2018.

Yang Guan received the B.S. degree from the School of Mechanical Engineering, Beijing Institute of Technology, Beijing, China, in 2017. He is working toward the Ph.D. degree with the School of Vehicle and Mobility, Tsinghua University, Beijing, China. His research interests include decision-making of autonomous vehicle, and reinforcement learning.

Yangang Ren received the B.S. degree from the Department of Automotive Engineering, Tsinghua University, Beijing, China, in 2018. He is working toward the Ph.D. degree with the School of Vehicle and Mobility, Tsinghua University, Beijing, China. His research interests include decision and control of autonomous driving, reinforcement learning and adversarial learning.

Shengbo Eben Li (Senior Member, IEEE) received the M.S. and Ph.D. degrees from Tsinghua University, in 2006 and 2009, respectively. He worked with Stanford University, University of Michigan, and University of California, Berkeley. He is currently a Tenured Associate Professor with Tsinghua University. He has author of more than 100 journal/conference papers, and the Co-Inventor of over 20 Chinese patents. His active research interests include intelligent vehicles and driver assistance, reinforcement learning and distributed control, optimal control and estimation, etc.

He was the recipient of Best Paper Award in 2014 IEEE ITS Symposium, Best Paper Award in 14th ITS Asia Pacific Forum, National Award for Technological Invention in China (2013), Excellent Young Scholar of NSF China (2016), Young Professorship of Changjiang Scholar Program (2016). He is currently the IEEE Senior Member and was a Associated Editor of IEEE INTELLIGENT TRANSPORTATION SYSTEMS MAGAZINE and IEEE INTELLIGENT TRANSPORTATION SYSTEMS, etc.

Qi Sun received the Ph.D. degree in automotive engineering from Ecole Centrale de Lille, France, in 2017. He did scientific research and completed the Ph.D. dissertation with CRIStAL Research Center, Ecole Centrale de Lille, France, between 2013 and 2016. He is currently a Postdoctor with the State Key Laboratory of Automotive Safety and Energy and the Department of Automotive Engineering, Tsinghua University, Beijing, China. His active research interests include intelligent vehicles, automatic driving technology, distributed control and optimal control.

Laiquan Luo received the M.S. degree from the Collage of Engineering, China Agricultrual University, Beijing, China, in 2019. He is currently an Algorithm Engineer with China Intelligent and Connected Vehicles (Beijing) Research Institute Co., Ltd. His research interests include computer vision and deep reinforcement learning.

Keqiang Li received the B.E. degree from Tsinghua University, Beijing, China, in 1985 and the M.S. and Ph.D. degree from Chongqing University, Chongqing, China, in 1988 and 1995, respectively. He is a Professor with Automobile Engineering, Tsinghua University. He has authored more than 90 papers and is Co-Inventor of 12 patents in China and Japan. His research interest includes vehicle dynamics and control for driver assistance system and hybrid electrical vehicle. He was a Senior Member of the Society of Automotive Engineers of China, and on the editorial boards of International Journal of Vehicle Autonomous Systems. He has received the Changjiang Scholar Program Professor and some awards from public agencies and academic institutions of China.