Safe Reinforcement Learning for Autonomous Vehicle Using Monte Carlo Tree Search

Shuojie Mo, Xiaofei Pei, and Chaoxian Wu

***Abstract*— Reinforcement learning has gradually demonstrated its decision-making ability in autonomous driving. Reinforcement learning is learning how to map states to actions by interacting with environment so as to maximize the long-term reward. Within limited interactions, the learner will get a suitable driving policy according to the designed reward function. However there will be a lot of unsafe behaviors during training in traditional reinforcement learning. This paper proposes a RL-based method combined with RL agent and Monte Carlo tree search algorithm to reduce unsafe behaviors. The proposed safe reinforcement learning framework mainly consists of two modules: risk state estimation module and safe policy search module. Once the future state will be risky calculated by the risk state estimation module using current state information and the action outputted by the RL agent, the MCTS based safe policy search module will activate to guarantee a safer exploration by adding an additional reward for risk actions. We test the approach in several random overtake scenarios, resulting in faster convergence and safer behaviors compared to traditional reinforcement learning.**

***Index Terms*— Reinforcement learning, autonomous vehicle, Monte Carlo tree search, decision-making.**

I. Introduction

REINFORCEMENT learning (RL) has been successfully used in sequential decision-making issues, such as chess [1], Atari games [2]. Deep RL (DRL) which combines RL with deep learning has the ability to handle realistic problems of continuous space. There are many researchers who have applied DRL method to autopilot decision problems [3], [4]. Little prior knowledge is needed in model-free RL algorithm due to the key idea of improving policies by interacting with the environment [5].

Though model-free RL approaches have achieved a great success in some areas, the safety issues caused by random exploration are still a big problem for safety-critical applications in real world, for example, collision avoidance. To get over those disadvantages, a subfield in RL called safe RL has gradually gained attractions in researchers. The purpose is to maximize the expectation return in which it is important to respect safety constraints. Previous safe RL is mainly consists of two parts: transformation of exploration process and optimization criterion [6].

For the first method in safe RL, additional variable, risk as an example, is considered when the exploration process is transformed. It's obvious that demonstrations can be used to calculate the additional information of state and action pairs by using approximate functions, such as neural network [7], Gaussian Processes [8], [9], etc. Such methods suffer the same shortcoming with imitation learning. Instead of demonstrations, unsafe actions can be shielded in advance and safe action will be carried out as a substitution by providing initial knowledge such as traffic rules, basic rule-based driving models, etc. [7], [10]–[12]. However, disadvantage still exists, where the agent does not know which action to shield when unexpected state occurs.

In optimization criterion-based safe RL, one way is to add constrains to criterion, where agent maximizes the expected return while keeping some values lower than designed bounds [13], [14]. The constrained criterion is applied in Constrained Markov Processes (CMDP), in which the feasibility is rarely guaranteed during learning in practice. In many studies, the optimization criterion is modeled as a combination of original return and risk, where risk can be defined various forms. Elfwing and Seymour introduced a MaxPain algorithm where two Q-values are used to estimate action-value and risk-value independently and then linear combined for final optimization [15]. The variance of the expected return can also be added to the optimization criterion, because higher variance implies more instability and in other words more risky [16].

In autonomous vehicle decision-making area, traditional rule-based is the most widely used method because of its simple structure, high practicality and technical maturity. The most well-known rule-based decision-making system can be found in DARPA Urban Challenge, where the Finite State Machine is implemented to decide how to drive in the urban scenario [17], [18]. However, with the rapidly increasing complexity of the driving scenario and the uncertainty from the imperfect perception system, the traditional rule-based systems cannot deal with such problems unless a perfect rule is designed to manage to be implemented in all the scenarios despite the existence of sensor error, which is impossible for now. Imitation learning is frequently utilized to learn policies from expert demonstrations for faster and safer learning in

Manuscript received July 7, 2020; revised November 9, 2020; accepted January 27, 2021. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 51505354. The Associate Editor for this article was J. Sanchez-Medina. *(Corresponding author: Xiaofei Pei.)*

Shuojie Mo is with the Department of Automotive Engineering, Wuhan University of Technology, Wuhan 430070, China (e-mail: moshuojie@whut.edu.cn).

Xiaofei Pei is with the Hubei Research Center for New Energy and Intelligent Connected Vehicle, Wuhan University of Technology, Wuhan 430070, China (e-mail: peixiaofei7@whut.edu.cn).

Chaoxian Wu is with the Hubei Key Laboratory of Advanced Technology for Automotive Components, Wuhan University of Technology, Wuhan 430070, China (e-mail: edward@whut.edu.cn).

Digital Object Identifier 10.1109/TITS.2021.3061627

autonomous driving [19]–[21]. However, expert demonstrations need to collect huge amount of data in advance, which is time consuming. Besides it cannot deal with unexpected occasions when no such demonstrations are imitated.

RL methods train the agent through interactions between agent and environment without the need of huge amount of data. Model-free algorithm, DQN for example, can be trained to make lane changes without the need of hand-crafted features or the model of environment [22], [23]. Researchers in Berkeley use several RL approaches to solve the decision-making problem with a bird-view input representation under the challenging roundabout scenario [24]. Continuous outputs including steers and actions can also be calculated by RL agent to finish missions such as lane keeping [25], lane changing [26], etc. The success of RL methods come from the trial and error learning, which also results in the disadvantage of traditional RL methods. Actions no matter safe or unsafe will be chosen by RL agent during exploration, leading to unsafe states and a low sampling efficiency.

In this paper, we propose a safe reinforcement learning framework combined with model-free reinforcement learning algorithm and Monte Carlo Tree Search (MCTS) method to improve the safety and exploration efficiency. Safe exploration module in safe RL consists of two parts: state estimation module and safe policy search module. Safe policy search module will be activated to find a safer action using MCTS when the state and action pair is estimated to be risky by the state estimation module after one-step prediction. No labelled data or prior expert knowledge is needed for safe action searching. The proposed safe RL framework is trained and tested in a two-lane overtaking scenario.

II. Background

A. Deep Q-Learning

The idea of reinforcement learning came from the nature of learning. During interacting with the environment, the agent gradually improves the action policy by exploring which actions yield the most reward. The basic mathematical framework of reinforcement learning is Markov decision process (MDP). A MDP is defined by a tuple of (S, A, P, R, γ), where S is a state space, A is a finite action space, R is the reward function, $\gamma \in [0, 1]$ is the discount factor, and P is the state transition probability which stands for the dynamics of the MDP. Q-Learning estimates the expected reward when taking action a in state s under a policy π. Such estimates of state and action pairs we call Q-values, which can be learned iteratively by updating the current Q-value estimate towards the observed reward and the Q-value of the successor state with implement of Bellman optimality equation.

$$Q(s,a) = Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right) \quad (1)$$

Deep Q-Learning (DQN) solve the estimate of high-dimensional state space problem by function approximation using deep neural network. Mnih et al firstly proposed experience replay memory and target network to solve unstable problem when a nonlinear function approximator is used to represent the Q-value [2]. Replay memory $D_t = \{e_1, e_2, \ldots, e_t\}$ is a pool where samples of sequence information $e_t = (s_t, a_t, r_t, s_{t+1})$ stored. It can remove correlations in the observation sequence thus greatly improving the performance. Target network is a historical copy of the main network with the same architecture. The parameters of target network are copied every t steps from main network and keep fixed on all others. DQN algorithm uses samples (or minibatches) of experience $(s, a, s', r) \sim U(D)$, drawn with a specific sampling way from the pool, for example. The algorithm update at iteration i uses the following loss function:

$$L_i(\theta) = E_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta^-) - Q(s,a;\theta)\right)^2\right] \quad (2)$$

in which γ is the discount factor, θ are the parameters of the main network and θ^- are the parameters used to compute the target Q-value in target network.

The parameters θ of the Q-values network is updated by the following gradient:

$$\theta_{i+1} = \theta_i + \alpha \nabla_\theta L_i(\theta) \quad (3)$$

B. Long Short Term Memory

Long short term memory (LSTM) is a kind of recurrent neural networks (RNN), which is specially proposed to solve the long-term dependency problem of RNN [27]. It has great power in processing time series data due to the particular LSTM structure and has been widely used in recent years. A standard LSTM block has three gates (input, forget and output), which is designed to control the data flow, deciding what to be remembered for a long time and what to be forgotten.

C. Deep Recurrent Q-Learning

Deep Recurrent Q-Learning (DRQN) is a model-free RL algorithm based on Deep Q-Learning. The key of this algorithm is adding recurrence to a Deep Q-Network by replacing the first fully-connected layer with a Long Short Term Memory (LSTM) layer [28]. It uses the same method with DQN to update the parameters of the neural network. The DRQN algorithm can better solve the incomplete and noisy state information benefiting from the great ability of sequence problem handling of LSTM.

D. Monte Carlo Tree Search

Monte Carlo tree search (MCTS) is a tree based search method combined with the generality of random sampling [29], [30]. It showed the great power in finding the best policy in exhaustive search problems after its breakthrough performance in AlphaGo [1]. A search tree is iteratively generated until the predefined limit with numerous nodes corresponding to states and actions. The values of nodes are estimated by many simulations. A tree policy is used to balance the exploration and exploitation problem and to find the most valuable nodes in the tree.

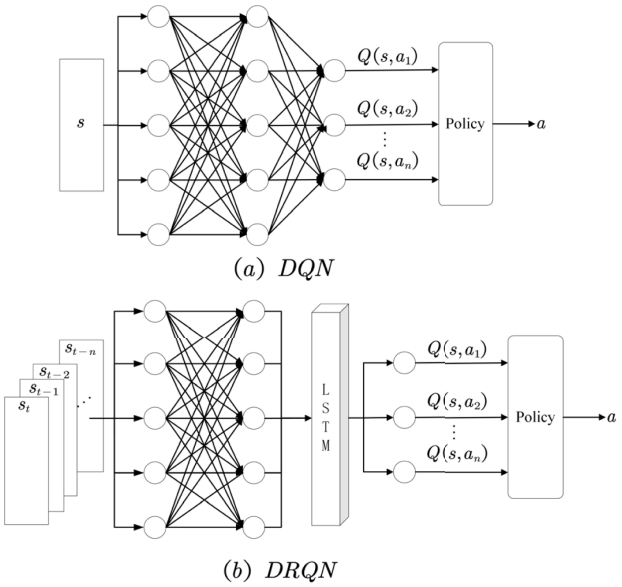

Fig. 1. Structure of DQN and DRQN.

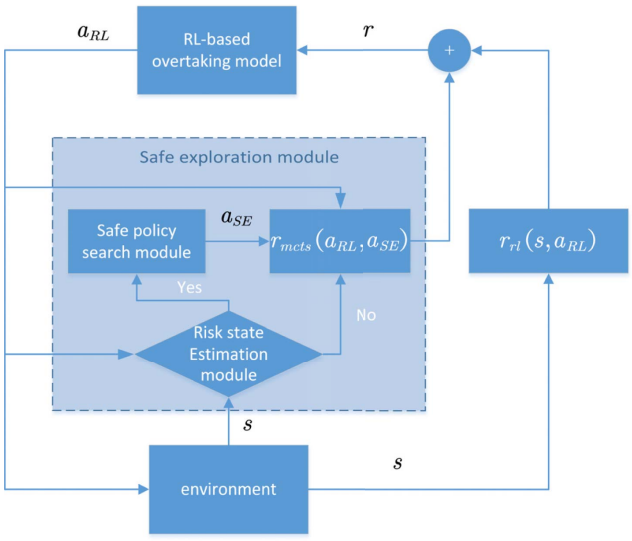

Fig. 2. Framework of the proposed safe reinforcement learning.

In the basic MCTS algorithm, there four steps in per iteration: selection, expansion, simulation and backpropagation. MCTS starts from the root node on the basis of information of the initial state, and a tree policy is utilized to choose a most valuable expandable node recursively in the selection process. Then child nodes are added to expand the tree using the available actions. The values of new added nodes are estimated by simulation and backpropagation process. A simulation is executed according to the default policy until the node terminates and get an outcome. The result will be back-propagated and update the value of the selected nodes.

The most commonly used selection function in MCTS is upper confidence bounds applied to trees (UCT), proposed by Kocsis and Szepesvári with the idea of combining tree search with UCB1 select policy for bandits [31]. It's recommended to choose unexplored actions if there is still any unexplored one, otherwise which maximizes UCT function with:

$$UCT(a) = v(s,a) + C\sqrt{\frac{InN(s)}{n(s,a)}} \tag{4}$$

where $n(s,a)$ represents the times action a is chosen at state s, and $N(s) = \sum_a n(s,a)$. The tradeoff constant of exploration and exploitation is represented by C. $v(s,a)$ is the estimated value of the node (s,a).

III. Approach

This section mainly describes the MDP modeling of lane-change decision-making problem of autonomous vehicle and the safe reinforcement learning method.

A. Safe Reinforcement Learning

In order to avoid the shortcomings of traditional reinforcement learning caused by random exploration, safe reinforcement learning method adds a safe exploration module based on Monte Carlo tree search (MCTS). The safe exploration module consists of two parts: risk state estimation module and safe policy search module based on MCTS. The framework of the proposed safe reinforcement learning approach is showed in Fig.2. Additional rewards r_{mcts} calculated by safe exploration module will be added to the original purpose-driven reward r_{rl} when the action a_{RL} generated by RL agent is unsafe and different from the search action a_{SE} using MCTS algorithm in safe policy search module.

1) Risk State Estimation: One-step prediction is obtained according to the action from RL agent and the current state and then the risk is estimated based on headway distance d_{gap} and time-to-collision (TTC) t_{ttc}. Safe headway distance $d_{gap}^{safe} = v_{ego} t_{thw}^{safe}$ is used for safety restrictions at lower speeds, in which we set $t_{thw}^{safe} = 1$s this time. Safe TTC $t_{ttc}^{safe} = 1.5$s is mainly applied to the collision safety limit in case of high speed emergency [32].

$$\begin{aligned} d_{gap} &= x_{\text{front}} - x_{\text{ego}} \geq d_{\text{gap}}^{\text{safe}} \\ t_{\text{ttc}} &= \frac{d_{\text{gap}}}{v_{\text{ego}} - v_{\text{front}}} \geq t_{\text{ttc}}^{\text{safe}} = 1.5\text{s} \end{aligned} \tag{5}$$

where x_{front}, x_{ego} are the positions of the front vehicle and the autonomous vehicle respectively. And v_{front}, v_{ego} are the velocity respectively.

If the predicted state is not satisfied with both headway distance and TTC restrictions, the safe policy search module activates to search the relatively safer actions. MCTS algorithm chooses the best action a_{SE} according to the UCT value of the node.

2) Safe Policy Search: To obtain a safer behavior relatively, MCTS algorithm is utilized in safe policy search module. Nodes consisting of state, action, as well as reward will be created in a tree through simulation, and best node will be chose based on the UCT values.

In this paper, safe policy search module tries to find a driving policy in a two-lane overtaking scenario without collisions or even risky behaviors. We design a more strict constraint during the simulation procedure in MCTS using

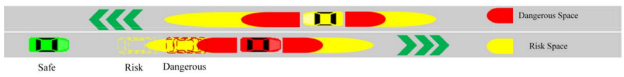

Fig. 3. Different levels of safety in a two-lane overtaking scenario.

a smaller time headway $t_{gap}^{\min} = 0.5$ and TTC $t_{ttc}^{\min} = 0.5$ compared with those in risk estimation module. When the autonomous vehicle is not satisfied with the minimum safety constraints, in other words, will be in dangerous state space (colored with red in Fig.3), the simulation will stop and return a negative terminal reward for subsequent back-propagation.

Several reward functions are set for different simulation termination:

Low-speed reward: Driving at a lower speed than front car can be safe in this scenario, but there is no meaning for the safe overtaking purpose in this paper. So when the average speed of autonomous vehicle is low than 3m/s, simulation in MCTS will be interrupted and return a negative reward $r_{low} = -2$.

Minimum safety reward: Minimum safety constraints consisting of TTC and time headway are used for early simulation stopping instead of a collision. When simulation stops for not meeting the minimum safety constraints, negative rewards will get as follows:

$$r_m = \begin{cases} -2 & \text{if } t_{\text{gap}} \le t_{\text{gap}}^{\min} \\ -\dfrac{1}{t_{ttc}} & \text{if } t_{ttc} \le t_{ttc}^{\min} \end{cases} \tag{6}$$

Overtaking reward: The only positive reward $r_o = 10$ will be obtained when the autonomous vehicle successfully overtaking the front vehicle without failure to meet the minimum safety constraints.

B. MDP Modeling

1) State Space: The state space S represents the features of driving information about the autonomous vehicle itself and the surrounding vehicle. In this paper, we take longitudinal distance into consideration only and the max perception range of autonomous vehicle is 150m. The state space is defined as follows:

$$\begin{aligned} s_t &= \left[\chi_t, \chi_{t-1}, \chi_{t-2}, \chi_{t-3}\right] \\ \chi_t &= (s_e, s_{ov}) \\ s_e &= (v_e, \text{Road}) \\ s_{ov} &= (\Delta x_i, v_i)_{i=1,2,\ldots\ldots,6} \end{aligned} \tag{7}$$

where χ_t is the abstract features of the environment at time t. In this paper, we use four continuous time-step features as state input s_t for DRQN algorithm. s_e and s_{ov} are features of autonomous vehicle and surrounding vehicles respectively. v_e is the velocity of autonomous vehicle and *Road* is the current lane ID. s_{ov} contains all the surrounding vehicle information in 6 directions including position and velocity information. We suppose that if there is no vehicle or vehicles out of perception range, 150m and -1m/s will be set as the default relative distance and velocity respectively.

2) Action Space: Action space A is defined as follows:

$$a = (\ddot{x}, LC) \tag{8}$$

where $\ddot{x}$ represents the longitudinal acceleration. We chose $(-3, -1, 0, 1, 2)$ as the discrete action values. *LC* is the high level lane change command. When the autonomous vehicle receives the high level command, low level controllers such as trajectory planner and control module will calculate velocity and headway for lane changing. In this paper, the next desired position can be calculated using a constant acceleration longitudinal dynamics with:

$$\begin{aligned} \dot{x}(t + \Delta t) &= \dot{x}(t) + \ddot{x}(t)\Delta t \\ x(t + \Delta t) &= \frac{1}{2}\ddot{x}(t)\Delta t^2 + \dot{x}(t)\Delta t + x(t) \end{aligned} \tag{9}$$

If action *LC* is chosen, the autonomous vehicle will finish the lane-change behavior in 2.5s$\sim$3s. Meantime, it keeps moving on with constant velocity of $\dot{x}(t)$.

3) Multi-Objective Reward Function: The main purpose of this simulation is to make the autonomous vehicle can overtake the front slow vehicle without collision, improve the driving efficiency, and ensure a certain degree of safety at the same time. The reward function consists of two parts: the purpose-driven reward r_{rl} and the safe exploration reward r_{mcts}. We incorporate a risk related reward function into the original reward function to make the agent safer. The final reward is $r = r_{rl} + r_{mcts}$.

The purpose-driven reward r_{rl} is the sum of three part: efficiency-related, safety-related and terminal-related ones:

a) Efficiency-related reward: The maximum velocity allowed on the road is set at 15m/s, and it is also set to the expected speed in the simulation. Linear function is used to encourage autonomous vehicles to adopt higher driving velocity:

$$R_v = \alpha_1(v_e - 15) \tag{10}$$

where $\alpha \in R^+$ is the reward adjustment coefficient, which is used to adjust the importance weight of reward function to the strategy.

b) Safety-related reward: Safety-related reward is mainly based on the TTC indicator. And relatively small additional penalty item C_{op} is added to prevent the self-driving vehicle from being in the opposite lane for a long time:

$$\begin{aligned} R_s &= \alpha_2 \min(0, t_{ttc} - 2.7) + C_{op} s_{ov} \\ C_{op} &= \begin{cases} -0.1 & \text{if in opposite lane} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{11}$$

c) Terminal-related reward: Terminal states will appear when there is a collision or overtaking, and the simulation ends. Different rewards will be given according to the terminal states:

$$R_t = \begin{cases} 10 & \text{if ter min } al = \text{overtake} \\ -10 & \text{if ter min } al = \text{collision} \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

The final purposed-driven reward function is the weighted sum of the above three reward. The weight coefficient needs

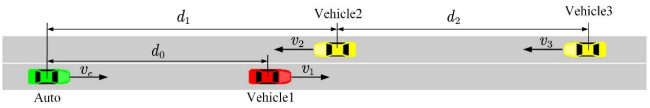

Fig. 4. Two-lane overtaking scenario.

to be designed according to the actual research purpose and fine adjustment is needed in the test.

And the safe exploration reward r_{mcts} is a fixed negative value unless there is no risk after one-step prediction or the action calculated by the RL agent is the same as the action came from safe exploration module:

$$r_{\text{mcts}}\left(a_{RL}, a_{SE}\right) = \begin{cases} 0 & \text{no risk or } a_{RL} = a_{SE} \\ -0.5 & \text{otherwise} \end{cases} \tag{13}$$

where a_{RL} is the action calculated by the RL agent and a_{SE} is calculated by the safe exploration module.

IV. Implementation And Results

A. Simulation Scenario

Overtaking is one of the most challenging behaviors in daily driving. This paper we test the proposed safe reinforcement learning approach in simulation scenario built in SUMO. Autonomous vehicle together with the other three normal vehicles is set on a two-lane road, shown in Fig.4. Slow vehicle drives in front in the same lane with the autonomous vehicle and other two vehicles run in the opposite lane. The distance between vehicles is randomly generated in each simulation. The autonomous vehicle needs to overtake the front slow vehicle with consideration of the driving conditions of vehicles in the opposite lane.

The green car Auto is controlled by the safe RL method with a constant initial velocity. The red Vehicle1 keeps a random-generated slow velocity all the time. The other two yellow vehicles drive in the opposite lane with a random initial velocity, and the driving strategy is based on the intelligent driver model (IDM) [33]. The IDM longitudinal motion control strategy is a continuous function of the velocity v, the gap s and the velocity difference $\triangle v$ to the leading vehicle:

$$\dot{v} = a_{\max}\left[1 - \left(\frac{v}{v_0}\right)^{\delta} - \left(\frac{s^*(v, \Delta v)}{s}\right)^2\right] \tag{14}$$

where $a_{\max} = 3\text{m/s}^2$ is the maximum acceleration of the normal vehicle, is acceleration exponent and $v_0 = 15\text{m/s}$ is the desired cruising velocity. $(s^*(v, \triangle v)/s)^2$ is the ratio between the desired minimum gap s^* and the actual gap s, which decides the tendency to brake when vehicle comes too close to the vehicle in front. And the desired minimum gap s^* is a function of velocity of own and velocity difference, given by:

$$s^*(v, \Delta v) = s_0 + \max\left(0, v_e T + \frac{v_e \Delta v}{2\sqrt{a \cdot b}}\right) \tag{15}$$

in which T, the safe time headway, is used for gap control.

TABLE I
Parameters of Two-Lane Scenario

Parameters	Symbol	Values
Length of road	l	1000m
Initial velocity of Auto	v_e	10m/s
Initial velocity of Vehicle1	v_1	[5 m/s,8 m/s]
Initial velocity of Vehicle2	v_2	[10 m/s,15 m/s]
Initial velocity of Vehicle3	v_3	[8 m/s,13 m/s]
Initial gap between Auto and Vehicle1	d_0	[30m,50m]
Initial gap between Auto and Vehicle2	d_1	[60m,200m]
Initial gap between Vehicle2 and Vehicle3	d_2	[40m,340m]
Time step	Δt	0.5s
Total simulation time	T	40s

TABLE II
Parameters Used in the Proposed Safe Reinforcement Learning

Parameters	Values
Episode	10000
Batch size	32
Length of input	5
Size of hidden layers	(100,50,50)
Size of replay memory	200000
Learning rate	1e-4
Discount rate	0.99
Exploration rate	0.3→0.1(annealed over 7e4 steps)
Iteration of MCTS	40
Exploration of MCTS	$\sqrt{2}/2$

Except for the autonomous vehicle Auto, the other vehicles do not change lanes during the simulation. Simulation ends after collision happens, or the autonomous vehicle runs at a low speed for a long time (invalid simulation results), or successfully overtakes the front vehicle. The simulation step is designed as 0.5s. The specific parameters during simulation can be seen in Table I.

B. Results

First we train the safe RL agent for 10000 episodes in the two-lane scenario with random initialization. The parameters we used for training in this paper can be found in Table. The information of surrounding vehicles during simulation is obtained by V2X communication system in SUMO platform without delays. A bigger iteration of MCTS algorithm will get a higher performance. In this paper, the maximum iteration of the MCTS algorithm in safe exploration module is set to 40 after comprehensive considerations of calculation efficiency and reliability.

Additionally, traditional DRQN-based lane-change model and rule-based MOBIL method [34] are taken as benchmarks and together tested in the random two-lane scenario. Eq.(16)

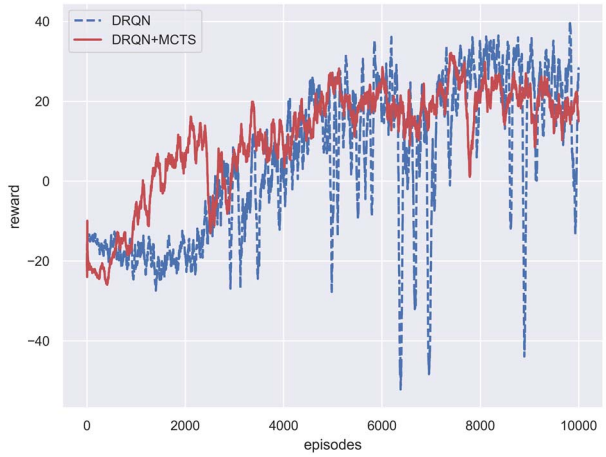

Fig. 5. Reward curve of DRQN and safe RL during training.

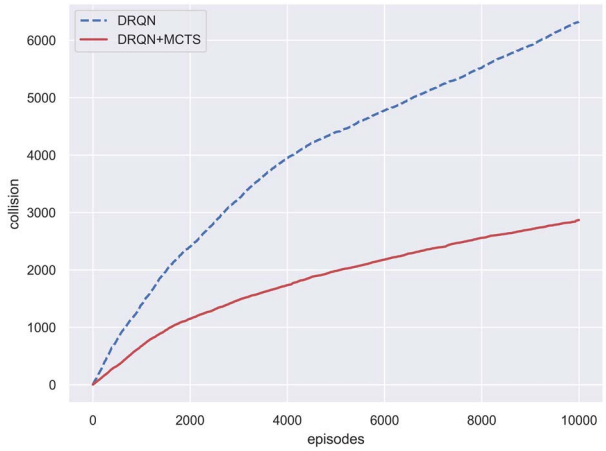

Fig. 6. Collison counts of DRQN and safe RL during training.

explains how MOBIL model decides when to change lanes:

$$\tilde{a}_c - a_c + p\left(\tilde{a}_n - a_n + \tilde{a}_o - a_o\right) > \Delta a_{\text{thread}} \tag{16}$$

where, a_c and $\tilde{a}_c$ are the accelerations of the host vehicle before and after lane change respectively. n, o presents the new follower and the old follower. And p is called politeness factor determining to which degree these vehicles influence the lane-changing decision. In this paper, we set the politeness factor $P = 0.2$ and the lane change threshold $\triangle a_{thread} = 0.1\text{m/s}^2$.

Fig.5 shows the reward curve of safe RL and DRQN. Comparison between safe RL and DRQN. Both methods can converge to a positive reward for ten thousands episode training. However, the safe RL agent has a faster convergence speed and a smaller variance by adding MCTS-based safe exploration module. And the collision counts during training can be seen in Fig.6. Obviously, safe RL method has a lower collision rate during trial and error learning. The additional negative safe exploration reward is well-worked to minimize the probability of risk behaviors.

Here we take the safe RL agent after training 1000 episodes as an example to show how the safe exploration module works.

TABLE III
COMPARISON OF DIFFERENT METHODS IN THE TEST

	MOBIL	DRQN	Safe RL
Overtake counts	226	257	300
Overtake rate	75.3%	85.7%	100%
Collision counts	0	43	0
Average speed, m/s	11.08	11.01	11.52

Green vehicle Auto is driving at a velocity of 11m/s after the slow red vehicle, which keeps a velocity of 7.9m/s. And the gap $s = 12$m. According to the current state and the acceleration action $a_{RL} = 1\text{m/s}^2$ outputted by the RL-agent (green arrow), the risk estimation module can obtain the next state using constant acceleration kinematic model and it's a risk state. The safe policy search module calculates a different lane change action a_{SE} (yellow arrow). The difference between two driving policy leads to a negative safe exploration reward thus reducing the possibility of acceleration action in this state in future training.

Finally the trained overtaking model will be tested for 300 rounds in the designed driving scenario. We collect all the TTC and time gap information during testing. Furthermore, the NGSIM human driving trajectory are taking into comparison. This dataset was collected in 2005 using multiple overhead cameras observing sections of highway. This data was taken over three sections of 15 minutes each and contains trajectories of roughly 5000 vehicles. Visual tracking techniques were used to extract vehicles trajectories from the image data at a rate of 10Hz [35]. The TTC and gap time values in NGSIM data are calculated. The distribution of gap time be found in Fig. 8. The test results of gap time result in safe RL method have a more concentrated distribution, most of which are in the relatively safe range of (1, 4). However, there are 2.19% results of DRQN method in the range of (0, 1), and 3.6% in NGSIM data, which is dangerous for driving. The same conclusion can be inferred from the distribution of TTC in Fig.9. According to the empirical safe TTC value of 1.5s in Van's research [36], the DRQN method is more risky. 4.77% TTC testing data of DRQN method lies between 0s and 1.5s, and 1.24% in NGSIM data, while the safe RL method is 0%. The velocity distribution is presented in Fig.10 in order to show the driving efficiency during testing. We set 15m/s as the maximum velocity in the overtaking scenario. The autonomous vehicle controlled by safe RL has a higher probability to achieve the limited maximum velocity compared to the vehicle controlled by the DRQN method.

For better comparisons, rule-based MOBIL lane change is also tested to show the performance of safe RL method. Overtake rate, collisions counts, average speed of Auto are chosen as the performance indicators of simulation result, in which the average speed of Auto is only calculated for successfully overtaking occasions. From the test results in Table III, it can be seen that the safe RL method obtains the highest overtaking success rate, while the traditional rule-based MOBIL lane-change model gets the lowest overtaking

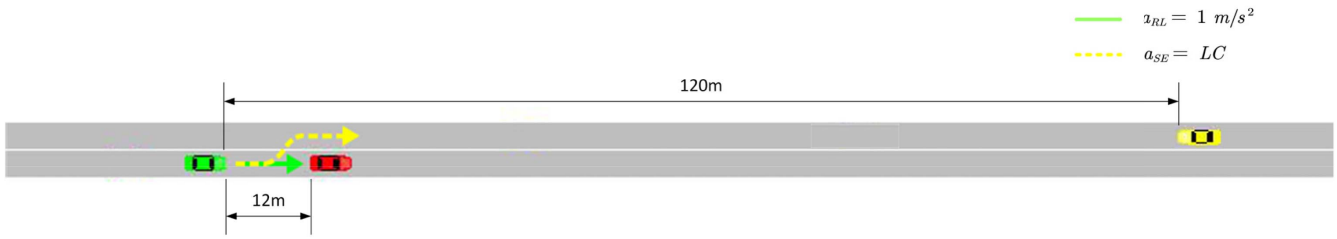

Fig. 7. Different driving policy between RL agent and MCTS.

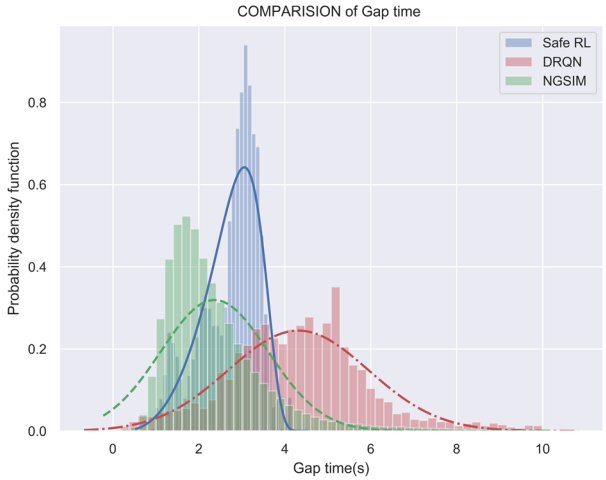

Fig. 8. Distribution of gap time in the two-way overtaking scenario.

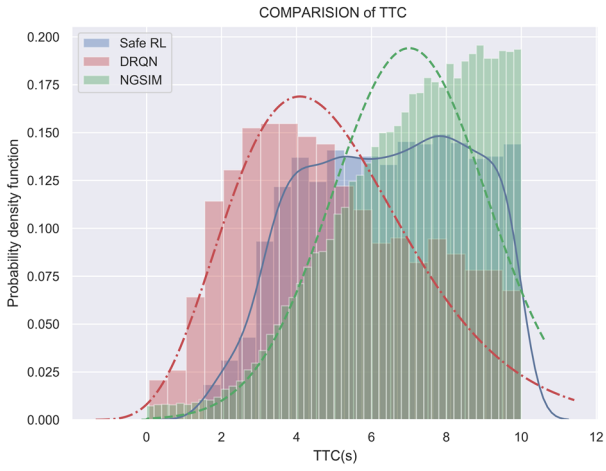

Fig. 9. Distribution of TTC in the two-way overtaking scenario.

rate due to the limited road length and time. Driving safety can be inferred from the collision counts, where both MOBIL and safe RL method get the best performance with 0 collision. However, the safety of DRQN method is not good enough after 10000 episodes' training because of the low training efficiency. As for the driving efficiency, we use average speed as an indicator, where the safe RL method gets the best performance

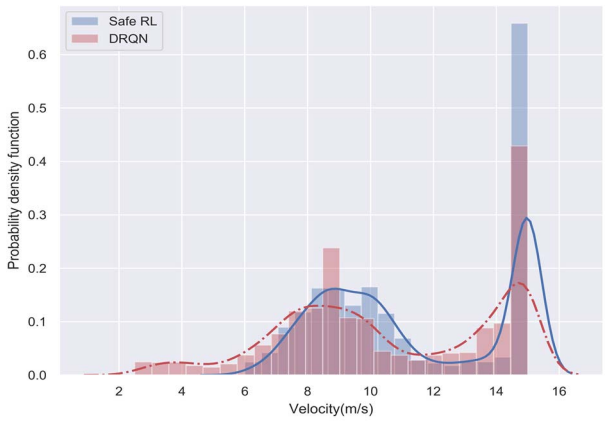

Fig. 10. Distribution of velocity in the two-way overtaking scenario.

among all three methods with a 3.97% speed increase over the MOBIL model and a 4.63% increase over DRQN agent.

V. Conclusion

This paper addresses decision making problem for the two-lane overtaking scenario using a safe reinforcement learning algorithm. It combines a traditional reinforcement learning algorithm with a search-based Monte Carlo tree search algorithm, in which an additional negative reward is added to actions that may involve risky driving behavior, thus speeding up the convergence of the algorithm.

Comparing the safe reinforcement learning algorithm with the rule-based MOBIL lane change model and the decision model based on the traditional reinforcement learning algorithm, it can be seen that the proposed safe reinforcement learning method has a larger increase in convergence speed, higher returns and better overtaking success rate for the same number of training episodes. Compared with the rule-based MOBIL lane change model, the driving efficiency and the opposite lane occupancy time are improved. The designed reward function can guide the agent to learn what we want the driving policy. In the future work, a combined decision-making system of rule-based methods and RL methods will be considered to make safer decision and improve training efficiency.

References

[1] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[2] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] X. Xu, L. Zuo, X. Li, L. Qian, J. Ren, and Z. Sun, "A reinforcement learning approach to autonomous decision making of intelligent vehicles on highways," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 10, pp. 3884–3897, Oct. 2020.

[4] T. Shi, P. Wang, X. Cheng, C.-Y. Chan, and D. Huang, "Driving decision and control for automated lane change behavior based on deep reinforcement learning," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 2895–2900.

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

[6] J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 1437–1480, 2015.

[7] A. Baheri, S. Nageshrao, H. E. Tseng, I. Kolmanovsky, A. Girard, and D. Filev, "Deep reinforcement learning with enhanced safety for autonomous highway driving," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Oct. 2020, pp. 1550–1555.

[8] Y. Sui, A. Gotovos, J. Burdick, and A. Krause, "Safe exploration for optimization with Gaussian processes," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 997–1005.

[9] J. Fan and W. Li, "Safety-guided deep reinforcement learning via online Gaussian process estimation," 2019, *arXiv:1903.02526*. [Online]. Available: http://arxiv.org/abs/1903.02526

[10] M. Mukadam, A. Cosgun, A. Nakhaei, and K. Fujimura, "Tactical decision making for lane changing with deep reinforcement learning," in *Proc. Workshop Mach. Learn. Intell. Transp. Syst.*, 2017, pp. 1–10.

[11] D. Chen, L. Jiang, Y. Wang, and Z. Li, "Autonomous driving using safe reinforcement learning by incorporating a regret-based human lane-changing decision model," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2020, pp. 4355–4361.

[12] M. Bouton, J. Karlsson, A. Nakhaei, K. Fujimura, M. J. Kochenderfer, and J. Tumova, "Reinforcement learning with probabilistic guarantees for autonomous driving," 2019, *arXiv:1904.07189*. [Online]. Available: http://arxiv.org/abs/1904.07189

[13] L. Wen, J. Duan, S. E. Li, S. Xu, and H. Peng, "Safe reinforcement learning for autonomous vehicles through parallel constrained policy optimization⋆," in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2020, pp. 1–7.

[14] M. Yu, Z. Yang, M. Kolar, and Z. Wang, "Convergent policy optimization for safe reinforcement learning," in *Proc. 33rd Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2019, pp. 3121–3133.

[15] S. Elfwing and B. Seymour, "Parallel reward and punishment control in humans and robots: Safe reinforcement learning using the MaxPain algorithm," in *Proc. Joint IEEE Int. Conf. Develop. Learn. Epigenetic Robot. (ICDL-EpiRob)*, Sep. 2017, pp. 140–147.

[16] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *J. Artif. Intell. Res.*, vol. 24, pp. 81–108, Jul. 2005.

[17] C. Urmson *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *J. Field Robot.*, vol. 25, no. 8, pp. 425–466, 2008.

[18] J. Leonard *et al.*, "A perception-driven autonomous urban vehicle," *J. Field Robot.*, vol. 25, no. 10, pp. 727–774, 2008.

[19] M. Bojarski *et al.*, "End to end learning for self-driving cars," 2016, *arXiv:1604.07316*. [Online]. Available: http://arxiv.org/abs/1604.07316

[20] F. Codevilla, M. Müller, A. Lopez, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1–9.

[21] M. Bansal, A. Krizhevsky, and A. Ogale, "ChauffeurNet: Learning to drive by imitating the best and synthesizing the worst," 2018, *arXiv:1812.03079*. [Online]. Available: http://arxiv.org/abs/1812.03079

[22] C.-J. Hoel, K. Wolff, and L. Laine, "Automated speed and lane change decision making using deep reinforcement learning," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 2148–2155.

[23] X. Li, X. Xu, and L. Zuo, "Reinforcement learning based overtaking decision-making for highway autonomous driving," in *Proc. 6th Int. Conf. Intell. Control Inf. Process. (ICICIP)*, Nov. 2015, pp. 336–342.

[24] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 2765–2771.

[25] A. Kendall *et al.*, "Learning to drive in a day," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 8248–8254.

[26] P. Wang, H. Li, and C.-Y. Chan, "Continuous control for automated lane change behavior based on deep deterministic policy gradient algorithm," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2019, pp. 1454–1460.

[27] F. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Comput.*, vol. 12, no. 10, pp. 2451–2471, 1999.

[28] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," 2015, *arXiv:1507.06527*. [Online]. Available: http://arxiv.org/abs/1507.06527

[29] R. Coulom, "Efficient selectivity and backup operators in Monte-Carlo tree search," in *Proc. Int. Conf. Comput. Games*, 2006, pp. 72–83.

[30] C. B. Browne *et al.*, "A survey of Monte Carlo tree search methods," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 1, pp. 1–43, Mar. 2012.

[31] L. Kocsis, C. Szepesvári, and J. Willemson, "Improved Monte-Carlo search," Univ. Tartu, Tartu, Estonia, Tech. Rep. 1, 2006.

[32] R. Horst, "A time-based analysis of road user behaviour in normal and critical encounters," TU Delft, Delft, The Netherlands, Tech. Rep., 1990. [Online]. Available: http://resolver.tudelft.nl/uuid:8fb40be7-fae1-4481-bc37-12a7411b85c7

[33] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 62, no. 2, pp. 1805–1824, 2000.

[34] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model MOBIL for car-following models," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 1999, no. 1, pp. 86–94, Jan. 2007.

[35] V. Alexiadis, J. Colyar, J. Halkias, R. Hranac, and G. McHale, "The next generation simulation program," *Inst. Transp. Eng. J.*, vol. 74, no. 8, pp. 22–26, 2004.

Shuojie Mo received the B.S. degree in mechanical engineering from the Zhejiang University of Technology, China, in 2018. He is currently pursuing the M.S. degree in vehicle engineering with the Wuhan University of Technology. His research interests include reinforcement learning and decision-making for autonomous vehicle.

Xiaofei Pei was born in Wuhan, Hubei, China, in 1985. He received the B.S. degree in mechanical engineering from the Wuhan University of Technology, Wuhan, in 2007, and the Ph.D. degree in mechanical engineering from the Beijing Institute of Technology, Beijing, China, in 2013. From 2011 to 2012, he was a Visiting Student with Rutgers University, New Brunswick, NJ, USA. Since 2016, he has been an Associate Professor with the Wuhan University of Technology. His research interests include vehicle active safety and autonomous vehicle.

Chaoxian Wu received the B.S. degree in vehicle engineering from the Wuhan University of Technology, China, in 2014, where he is currently pursuing the Ph.D. degree. He is also a Visiting Student with the Department of Mechanical Engineering, Virginia Tech, Blacksburg, VA, USA. His research interests include mechatronics design, vehicle brake systems, vehicle dynamics and control, and autonomous vehicle.