Motion Planner with Fixed-Horizon Constrained Reinforcement Learning for Complex Autonomous Driving Scenarios

Ke Lin, *Graduate Student Member, IEEE*, Yanjie Li, *Member, IEEE*, Shiyu Chen, Duantengchuan Li, Xinyu Wu, *Senior Member, IEEE*

***Abstract*—In autonomous driving, behavioral decision-making and trajectory planning remain huge challenges due to the large amount of uncertainty in environments and complex interaction relationships between the ego vehicle and other traffic participants. In this paper, we propose a novel fixed-horizon constrained reinforcement learning (RL) framework to solve decision-making and planning problems. Firstly, to introduce lane-level global navigation information into the lane state representation and avoid constant lane changes, we propose the constrained A-star algorithm, which can get the optimal path without constant lane changes. The optimality of the algorithm is also theoretically guaranteed. Then, to balance safety, comfort, and goal completion (reaching targets), we construct the planning problem as a constrained RL problem, in which the reward function is designed for goal completion, and two fixed-horizon constraints are developed for safety and comfort, respectively. Subsequently, a motion planning policy network (planner) with vectorized input is constructed. Finally, a dual ascent optimization method is proposed to train the planner network. With the advantage of being able to fully explore in the environment, the agent can learn an efficient decision-making and planning policy. In addition, benefiting from modeling the safety and comfort of the ego vehicle as constraints, the learned policy can guarantee the safety of the ego vehicle and achieve a good balance between goal completion and comfort. Experiments demonstrate that the proposed algorithm can achieve superior performance than existing rule-based, imitation learning-based, and typical RL-based methods.**

***Index Terms*—Autonomous driving, trajectory planning, constrained reinforcement learning, fixed-horizon constraint, policy learning.**

I. Introduction

THERE has been considerable interest in autonomous driving technology by both industry and academia in recent years [1–3]. This is mainly due to potential benefits that autonomous driving technology could bring to our daily lives, such as enhancing road safety, alleviating parking difficulties, improving daily commutes, and reducing emissions. Notably, behavioral decisions and trajectory planning have become critical components in autonomous driving [4].

In general, decision-making and planning algorithms can be divided into two categories: traditional rule-based methods [5–13] and learning-based methods. Moreover, the latter can be divided into imitation learning (IL) based approaches [14–22] and reinforcement learning (RL) based algorithms [23–28].

Rule-based methods: In traditional rule-based autonomous driving, behavioral decision-making and planning problems are usually solved separately. For behavioral decision-making, lane change decision is a key component as a lateral decision model. Common lane change models include SUMO's lane change model [5], MOBIL model [6], and personalized driver model [7]. Particularly, in SUMO's lane change model, according to the urgency of the decision, cooperative lane-changing, tactical lane-changing, and regulatory lane-changing are considered to cope with different situations. In addition, the Intelligent Driver Model (IDM) [8] is a classic single-lane acceleration control model (car flow model), which is often used for longitudinal decisions in typical autonomous vehicles. Other car flow model includes [9, 10]. To achieve intelligent behavioral decision-making, a series of rules are often designed according to the experience of experts, and the finite state machine can be used to model these rules. Kurt et al. [11] proposed a hierarchical layout driverless control system based on a finite state machine, which successfully controlled a car to complete the DARPA[1] urban challenge. On the other hand, for trajectory planning, given lanes and static obstacles around the ego vehicle, the trajectory can be obtained using numerical optimization [12], heuristic search algorithms [13], and spline smoothing approaches.

IL-based methods: In the face of complex traffic scenarios, rule-based methods are usually not applicable. This is because it may be difficult to design effective and elaborate rules for complex scenarios. Therefore, IL-based methods have become popular in recent years. Chauffeurnet [15] is a deep learning model that learns a robust autonomous driving planning model by adding random perturbations to expert datasets. Taking rasterized images as input, Chauffeurnet successfully drives

This work was supported in part by the National Natural Science Foundation of China under Grant 61977019 and Grant U1813206; and in part by the Shenzhen Fundamental Research Program under Grant JCYJ20180507183837726, Grant JCYJ20220818102415033, and Grant JSGG20201103093802006. (*Corresponding author: Yanjie Li*)

Ke Lin, Yanjie Li, Shiyu Chen are with the Department of Control Science and Engineering, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China. (e-mail:*schris_lin@stu.hit.edu.cn; autolyj@hit.edu.cn; chenshiyu@stu.hit.edu.cn.*)

Duantengchuan Li is with the School of Computer Science, Wuhan University, Wuhan 430072, China. (e-mail: *dtclee1222@whu.edu.cn*)

Xinyu Wu is with the Guangdong Provincial Key Laboratory of Robotics and Intelligent System, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China, and also with the Guangdong-Hong Kong-Macao Joint Laboratory of Human-Machine Intelligence-Synergy Systems, Shenzhen 518055, China. (e-mail: *xy.wu@siat.ac.cn*)

[1]https://www.darpa.mil

a real car in a test facility. Different from the rasterized input in Chauffeurnet, Zeng et al. [16] proposed a planning network that directly takes the lidar and the original high-definition (HD) map as input. It can generate a safe planning result by sampling a set of diverse physically possible trajectories and choosing the safest one. Moreover, MP3 [17] does not use HD map information but directly utilizes lidar data to construct local map information. In MP3, an innovative neural motion planner is constructed by considering the uncertainty information of other vehicles. On the other hand, vectorized representation is another common state representation pattern in the field of autonomous driving. The most typical state construction method is VectorNet [18], in which maps, lanes, and all traffic participants are represented by a series of vectors. Compared with the rasterization input, the benefit of vectorized input is that it does not lose any representation accuracy. Based on the VectorNet, Scheel et al. proposed a motion planning model trained by an offline policy gradient method [19], which achieves excellent performance in complex autonomous driving scenarios. In addition, there are autonomous driving planning methods based on generative adversarial IL [21, 22], which have also achieved excellent performance in many complex scenarios.

RL-based methods: However, in the existing autonomous driving datasets used in IL-based methods, most of the data is collected during normal driving situations. For example, a large amount of data is collected when driving straight. There is not much data on challenging cases, like when a car brakes suddenly in front of the ego vehicle or a pedestrian crosses at a red light. This feature makes the existing IL-based methods unable to effectively deal with special corner cases. Thus, with the assistance of high-fidelity simulation environments, RL is introduced into behavioral decision-making and planning [23]. In RL, an agent can fully explore various situations in environments and obtain a planner that could maximize the cumulative rewards [29, 30]. Huang et al. [24] proposed a prior knowledge-based RL algorithm that achieved good performance in the scenarios of a left turn at intersections and roundabouts. Aiming at the problem of obstacle avoidance in urban scenes, Zhao et al. [25] developed an RL algorithm that utilizes a hybrid offline and online training technique, which successfully realizes efficient obstacle avoidance on urban roads. In order to ensure the stability and safety of the RL algorithm in trajectory planning, Gangopadhyay et al. [26] proposed an RL algorithm based on the control barrier functions (CBF) and control Lyapunov functions, which successfully controlled the vehicle to reach the target point in a simulation environment. Similarly, for the control problem with vehicle obstacle avoidance, Hu et al. [27] also proposed a model-based safety exploration method based on CBF. Other efficient RL-based planners have been summarized in [28].

Existing RL-based methods can only maximize the cumulative reward during the policy optimization process. In order to balance safety, comfort, and goal completion (reaching targets) in planning, it is usually necessary to carefully design the reward function in environments which is usually intractable. In many cases, agents may sacrifice safety or comfort to achieve better goal completion for higher rewards or become quite conservative for absolute safety. In this paper, to better balance the goal completion, safety, and comfort, we model the problem as a constrained RL, in which the reward item aims to make the ego reach the target as soon as possible, and the constraint components consider the safety, comfort, and violations of traffic rules. The overall optimization goal is that the planner needs to maximize the cumulative reward while satisfying all constraints (e.g., safety constraint and comfort constraint). As a consequence, we propose a fixed-horizon constrained RL-based planner for autonomous driving with complex scenarios. The main contribution of this work is summarized as follows:

- A fixed-horizon constrained RL framework is designed, in which the fixed-horizon cumulative cost is utilized to build the safety and comfort constraints. The planner trained by the proposed framework can achieve an excellent balance in goal completion, safety, and comfort for decision-making and planning tasks in autonomous driving.
- To implement constrained policy training, a planner network is designed, in which a graph network is applied to extract the features of lanes, traffic participants, etc. Then, a dual ascent policy learning method is developed, which enables the learned policy to maximize the cumulative reward while satisfying all the constraints.
- To introduce the global navigation information in lane state representation in the planner network, a constrained A-star algorithm is developed. It can be guaranteed that the planned lane-level global path will not change lanes constantly. In addition, the optimality of the constrained A-star algorithm is theoretically proved.
- Experiments demonstrate that the proposed algorithm can successfully perform tasks such as changing lanes, waiting for red lights, and turning right in a complex traffic scenario, and can reach the target point safely, comfortably, and quickly. In addition, compared with existing rule-based, IL-based, and RL-based approaches, the proposed method can achieve better performance in terms of goal completion, safety, and comfort.

The remainder of this paper is structured as follows. In Section II, we describe the preliminary of the constrained RL. In Section III, the fixed-horizon constrained RL-based planner is introduced. In Section IV, the proposed method is verified in simulation experiments. Comparisons with existing rule-base, IL-base, and RL-based approaches are also carried out. Finally, conclusions and future work are summarized in Section V.

II. Preliminary

A. Constrained RL

Generally, constrained RL problems are often modeled using a constrained Markov decision process (CMDP) [31]. A CMDP can be represented by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, J_{c_i}, C_i, \gamma)$, where $\mathcal{S}$ represents a state space and $\mathcal{A}$ denote an action space. $P(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ represents the probability of the next state $\boldsymbol{s}' \in \mathcal{S}$ occurring given the current state $\boldsymbol{s} \in \mathcal{S}$ and action $\boldsymbol{a} \in \mathcal{A}$. In addition, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward given by the environment on

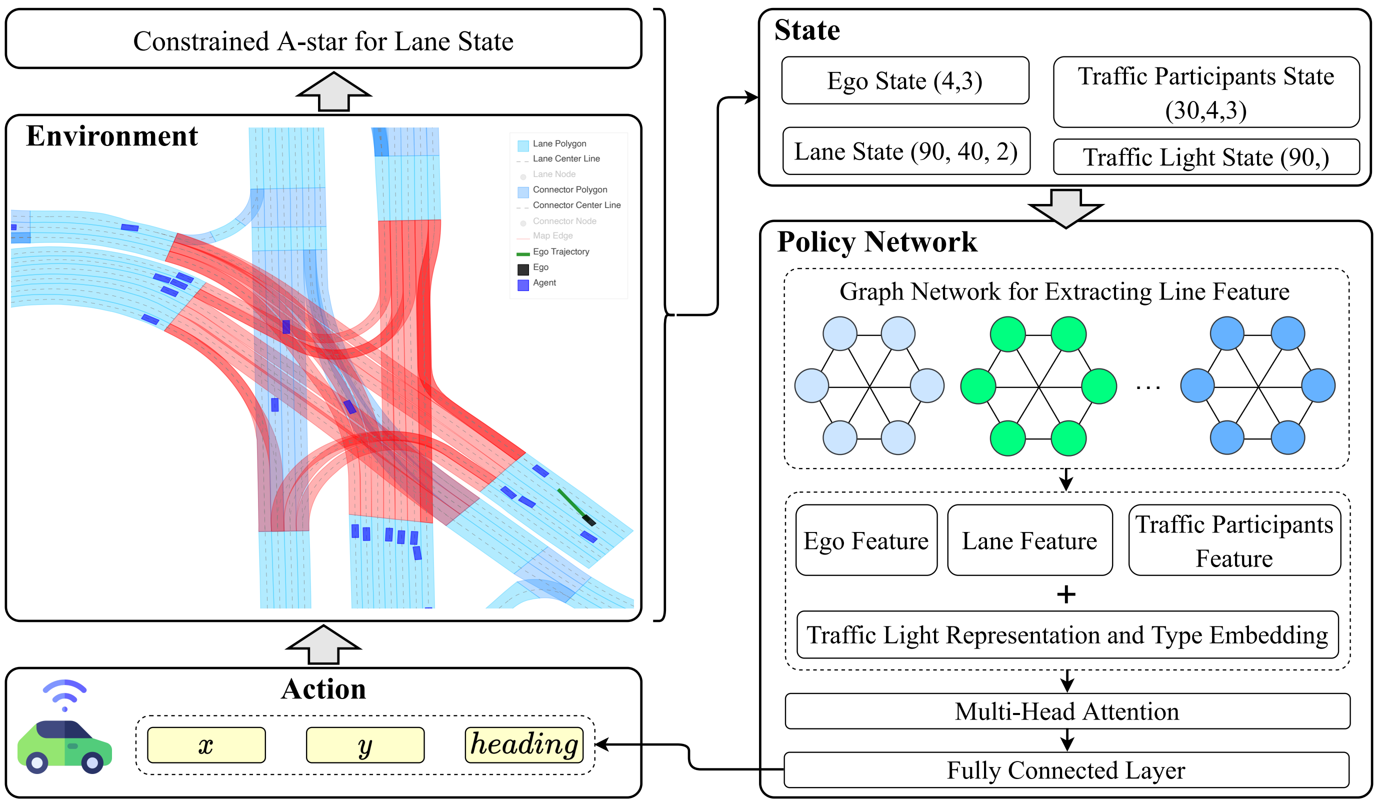


Fig. 1. An illustration of the proposed RL-based framework.

each state transition. J_{c_i} is a constraint function, which defines cost in a constraint and C_i is a cost threshold. $\gamma \in [0, 1]$ is a discount factor. A standard CMDP aims to find a policy π that can maximize the expected sum of rewards while satisfying all constraints to make the agent have risk-sensitive behavior. A CMDP problem is formally defined as

$$\begin{aligned} &\max_{\pi} \quad \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t)\right], \\ &\text{s.t.} \quad J_{c_i}(\pi) \le C_i, \quad i = 1, ..., n. \end{aligned} \tag{1}$$

Moreover, the state value function $V^{\pi}(\boldsymbol{s})$ and the state-action value function $Q^{\pi}(\boldsymbol{s}, \boldsymbol{a})$ are introduced to evaluate the performance of the policy. Specifically, they are defined as the discounted total reward under the policy π given the initial state and action. From the definition, the following equations hold [32]:

$$V^{\pi}(\boldsymbol{s}) = \sum_{\boldsymbol{a}} \pi(\boldsymbol{a}|\boldsymbol{s}) \sum_{\boldsymbol{s}'} P(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) \left[r + \gamma V^{\pi}(\boldsymbol{s}')\right], \tag{2}$$

$$Q^{\pi}(\boldsymbol{s}, \boldsymbol{a}) = r + \gamma \sum_{\boldsymbol{s}'} P(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) V^{\pi}(\boldsymbol{s}'). \tag{3}$$

III. METHODOLOGY

This section first describes the proposed constrained A-star search algorithm for global routing, which is used for lane state representation in the input of the planner network. Then, the state space, action space, reward, and cost design in our method are introduced. Subsequently, the proposed motion planning policy network is described, and the fixed-horizon constrained policy learning with a dual ascent algorithm is demonstrated. An illustration of the proposed framework can be seen in Fig. 1.

A. Constrained A-star for Lane-level Global Routing

In order to realize lane-level navigation for lane state representation, a directed graph is constructed based on the relationship among lanes in the HD map. Each node in the graph represents a lane, and the relationship among nodes includes *successor*, *left neighbor*, and *right neighbor*. In addition, the weight on the edge connecting each node is the distance between center points of the two lanes. Fig. 2 shows a lane graph near an intersection.

Given a starting lane node and a target lane node, the shortest path algorithm (e.g., Dijkstra's algorithm or A-star algorithm) can be directly used to obtain the optimal lane-level path with the shortest distance. Note that consecutive lane changes are uncomfortable and, in some areas, even against traffic regulations. However, when solving lane-level global routing problems, existing shortest path algorithms may give a solution with constant lane changes. To deal with this problem, we propose the constrained A-star algorithm. The path it plans does not change lanes constantly and can find the shortest path under the constraint of non-constant lane changes.

In normal A-star algorithms, the shortest path from the starting node to the target node can be obtained by iterative

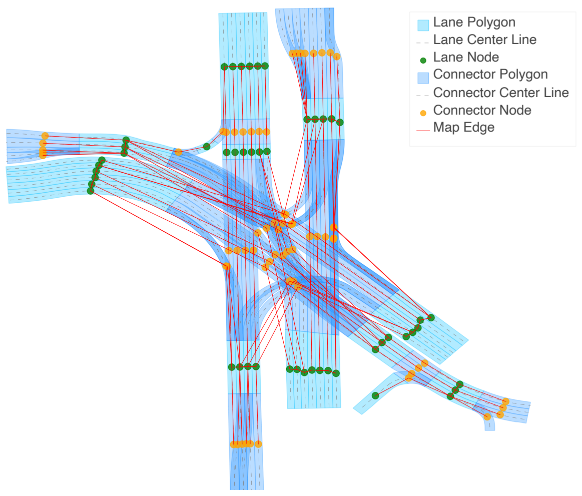


Fig. 2. A lane graph near an intersection.

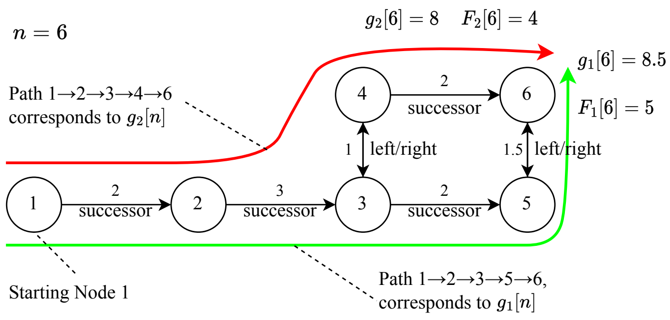


Fig. 3. An example to illustrate definitions of $g_1[n]$, $g_2[n]$, $F_1[n]$, and $F_2[n]$.

updates. Let $g[n]$ represent the distance from the starting node to node n in the iterative process, then $g[n]$ can be updated by

$$g[n] = \min_{m \in prev(n)} \{g[m] + G[m][n]\}, \tag{4}$$

where $prev(n)$ is defined as previous nodes of node n, and $G[m][n]$ indicates the weight of the edge $E(m,n)$. In our constrained A-star algorithm, we first give the following definition:

Definition 1. *$g_1[n]$ and $g_2[n]$ are defined as the distance from the starting node to node n in the iterative process, where the attributes of the last edge on the path represented in $g_1[n]$ and $g_2[n]$ are left (or right) neighbor and successor, respectively.*

Definition 2. *$G_1[i][j]$ and $G_2[i][j]$ are defined as the weight on the edge connecting two adjacent nodes i to j, where the relationships between i and j are left (or right) neighbor and successor, respectively. When i and j are not connected, $G_1[i][j] = \infty, G_2[i][j] = \infty$.*

Definition 3. *$F_1[i]$ and $F_2[i]$ are defined as the parent node of node i in the iterative process, where the relationship between $F_1[i]$ and i is left (or right) neighbor, and the relationship between $F_2[i]$ and i is successor.*

To further explain the meaning of the above definitions, an example is shown in Fig. 3. Let the starting node be 1 and $n = 6$. Then, $g_1[6]$ is the shortest path distance from node 1 to node 6, where the attribute of the last edge is left neighbor, and the corresponding path is $1 \to 2 \to 3 \to 5 \to 6$. At this time, $F_1[6]$ is the previous node of node 6 in the path, so $F_1[6] = 5$. Similarly, $g_2[6]$ is the shortest path distance from node 1 to node 6, where the attribute of the last edge is successor. The corresponding path is $1 \to 2 \to 3 \to 4 \to 6$ and $F_2[6] = 4$.

Then, in the iterative process, we can update $g_1[n]$ and $g_2[n]$ with the following form:

$$g_1[n] = \min_{m \in prevn(n)} \{g_2[m] + G_1[m][n]\}, \tag{5}$$

$$g_2[n] = \min_{m \in prev(n)} \{\min\{g_1[m], g_2[m]\} + G_2[m][n]\}, \tag{6}$$

where $prevn(n)$ represents previous nodes of node n, which satisfies that the attribute of the edges connecting them and node n is *successor*. Notably, using (5) to update g_1 can ensure that there will be no path with constantly changing lanes. Thus, we can give the constrained A-star algorithm in Algorithm 1.

After obtaining g_1, g_2, we get the final path based on F_1, F_2. It can be shown that the case of constant lane change is avoided when the path is constructed in Algorithm 1. For the shortest path $path^*$, (5) and (6) must hold, which can be viewed as the optimality equation. Otherwise, there will be a shorter path. Therefore, when the heuristic function is defined as $h = 0$, the path obtained according to Algorithm 1 must be optimal under the constraint of non-constant lane changes. Detailed proof of the optimality of Algorithm 1 is shown in Theorem 1.

Theorem 1 (optimality). *Let $\delta[n]$ be the shortest path distance without constant lane changes from the starting node to node n in the lane graph. Also, let $g[n] = \min\{g_1[n], g_2[n]\}$, where $g_1[n], g_2[n]$ are obtained by the constrained A-star Algorithm. Then, when the heuristic function satisfies $h = 0$, we have $g[n] = \delta[n]$.*

Proof. We prove it by induction. Denote X as the closed node set and the starting node ID as 1. When $|X| = 1$, we denote X as X_1. Then, we have $X_1 = \{1\}$, and $g[1] = 0 = \delta[0]$. When $|X| = k$, let us assume $g[x] = \delta[x], x \in X_k$. Then, when $|X| = k+1$, let us denote $X_{k+1} = X_k \cup \{u\}$. Thus, we only need to prove that $g[u] = \delta[u]$.

We prove it by contradiction. Suppose exist the shortest path W without constant lane changes from node 1 to node u, whose length is $l(W)$, that satisfies $l(W) < g[u]$.

In the path W, denote $E(p,q)$ as the first edge that leaves X_{k+1}. Then, one can have

$$l(W_p) + l(p,q) \le l(W),$$

where W_p is the sub-path of W, who connects node 1 to node p. According to the assumption when $|X| = k$, $g[x], x \in X_k$, $g[p]$ is the shortest path distance without constant lane changes, one can obtain $g[p] \le l(W_p)$. Then we have

$$g[p] + l(p,q) \le l(W).$$

Considering the fact that p and q are adjacent and in the constrained A-star Algorithm g_1 and g_2 are updated by minimize operations, we can get $g_1[q] \le g_2[p] + l(p,q)$ when the attribute of $E(p,q)$ is successor, and $g_2[q] \le g[p] + l(p,q)$. In addition,

Algorithm 1 Constrained A-star Algorithm

Input: A graph G_1, and a graph G_2. Let starting node ID as 1 and target node ID as N. Two distance arrays g_1 and g_2. A heuristic function h. Two parent arrays F_1 and F_2. An open node set $\Omega = \{1, ..., N\}$.

Procedure:

```
1: Initialize $g_1[1] = 0, g_2[1] = 0$.
2: Initialize $g_1[k] = \infty, g_2[k] = \infty, k = 2, ..., N$.
3: while $\Omega$ is not empty do
4:     Let $m = \min_{k\in\Omega} \{\min\{g_1[k], g_2[k]\} + h(k, N)\}$.
5:     Get all next nodes of node $m$ as a set $B$.
6:     for $n$ in $B$ do
7:         if the attribute of edge $E(m, n)$ is successor then
8:             Let $p = \min\{g_1[m], g_2[m]\} + G_2[m][n]$.
9:             if $p < g_2[n]$ then
10:                Let $g_2[n] = p$ and $F_2[n] = m$.
11:            end if
12:        else
13:            Let $p = g_2[m] + G_1[m][n]$.
14:            if $p < g_1[n]$ then
15:                Let $g_1[n] = p$ and $F_1[n] = m$.
16:            end if
17:        end if
18:    end for
19:    Remove $n$ from $\Omega$.
20: end while
21: Let $path = []$, $le = 1$ and $n = N$.
22: while $n$ is not staring node do
23:     if $le == 1$ and $g_1[n] < g_2[n]$ then
24:         Let $n = F_1[n]$ and $le = 0$.
25:     else
26:         Let $n = F_2[n]$ and $le = 1$.
27:     end if
28:     Add $n$ to $path$.
29: end while
30: Reverse $path$.
```

Output: $\min\{g1[N], g2[N]\}, path$.

the next point u selected in the constrained A-star algorithm is based on the smallest g, then we have $g[u] \leq g[q]$. Finally, one can get

$$\begin{aligned} g[u] \leq g[q] &= \min\{g_1[q], g_2[q]\} \\ &\leq \min\{g_2[p] + l(p, q), g[p] + l(p, q)\} \\ &= g[p] + l(p, q) \leq l(W) \\ &< g[u], \end{aligned}$$

which makes a contradiction. Therefore, there is no such path W that satisfies $l(W) < g[u]$. □

B. State Space and Action Space Design

With the constrained A-star algorithm, the lane state in state space can be constructed based on the resulting global routing information. To fully represent all the information that the agent can perceive, there are four main parts in the state space, including ego vehicle, traffic participants, lane, and traffic light and type representation.

1) Ego Vehicle: The state of the ego contains the position and the heading angle $(x, y, heading)$ of the current moment and the three historical moments. Thus, a matrix with the shape of (4,3) is used to represent the state of the ego. The position and heading angle are in the coordinate system based on the ego vehicle at the current moment.

2) Traffic Participants: The state of the traffic participants is similar to the ego vehicle. In our setting, up to 30 traffic participants within 75 meters of the ego vehicle are considered. Thus, a tensor with the shape of (30, 4, 3) is used to represent the state of traffic participants.

3) Lane: To make full use of global navigation information, only lanes that exist in the planned path by the constrained A-star are included. In addition, up to 30 lanes within 100 meters of the ego vehicle are taken into account. The representation of a lane consists of three parts: the centerline of the lane, and the left and right sidelines of the lane. In the setting of this article, 40 points are used to represent a line (centerline, left line, or right line), and each point is represented by the position (x, y). Thus, all centerlines can be represented by a tensor whose shape is (30, 40, 2). The entire lanes can be represented with 3 tensors (center, left, and right lines), and the size of each tensor is (30, 40, 2). The coordinates of these points are based on the ego vehicle coordinate system.

4) Traffic Light and Type Representation: In order to represent the traffic lights and types of traffic participants and different lane lines, an extra type encoding is used, including *tp_car*, *tp_bike*, *tp_pedestrian*, *tl_red*, *tl_yellow*, *tl_green*, *tl_none*, *line_solid*, and *line_dashed*, in which tp means traffic participants and tl means traffic light. Considering that there are at most 30 traffic participants, 30 lane centerlines, 30 left lanes, and 30 right lanes, a total of 120 elements need to be coded. Therefore, a vector with a length of 120 is used to represent their type.

5) Action Space: The main advantage of RL is the ability to make intelligent decisions, so the control part of the vehicle is not considered. Therefore, the action is defined as the position and heading angle of the ego vehicle after 0.1 seconds on the ego vehicle coordinate system.

C. Reward and Cost Design

The algorithm proposed in this paper aims to make the ego drive to the target lane while ensuring the safety and comfort of the vehicle. Therefore, the goal completion and the speed of the agent are considered in the design of the reward. Safety and comfort of the agent are taken into account in the design of the cost.

1) Reward for Reaching Target: In order to enable the ego vehicle to reach the given target lane, the reward for reaching the goal r_t is defined as

$$r_t = \Delta x \cdot x_t^r + \Delta y \cdot y_t^r + 0.25 \cdot (\Delta x^2 + \Delta y^2), \quad (7)$$

where $(\Delta x, \Delta y)$ is the ego's movement distance along the two axes. (x_t^r, y_t^r) is the tangent direction of point p_t on the planning path τ_t, where p_t is the closest point to the ego on τ_t. τ_t is the standard trajectory which is defined as follows: τ_t is the lane centerline when no lane changes are needed, or

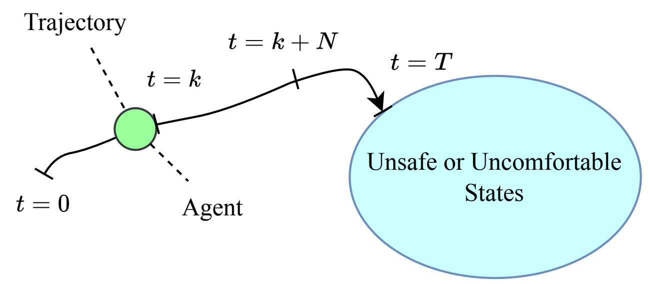


Fig. 4. Intuition about why the fixed-horizon constraints can alleviate the conservativeness of the agent in the learning phase.

τ_t is the straight line between the start point and end point of the two lanes when a lane change is needed. Thus, the agent will be rewarded for driving along the lane and moving fast. To avoid speeding of the ego vehicle, the values of $\Delta x, \Delta y$ are truncated in our environment.

2) Safety Cost (collision): In order to avoid collisions between the ego vehicle and traffic participants and avoid violation of traffic rules, an additional cost value is output in the environment at each time to indicate the safety of the agent. The safety cost c_t^s is defined as

$$c_t^s = \mathbb{1}\{\text{collision or violation of traffic rules}\}, \tag{8}$$

where $\mathbb{1}\{\}$ is the indicator function.

3) Comfort Cost (jerk): The previous reward design and safety cost design have been able to make the agent safely reach the target point. In addition, in order to ensure sufficient comfort during driving, an additional comfort cost c_t^c is set in this environment, which is defined as

$$c_t^c = \mathbb{1}\left\{acc_t < C_a\right\}, \tag{9}$$

where acc_t represents the acceleration of the ego vehicle at time t. C_a is a threshold for acceleration, which is set to 4 to avoid uncomfortable situations.

D. Motion Planning Policy Network

The essence of the policy network is to learn a mapping from the state space to the action space, so that the agent can safely and comfortably drive to the target point. Inspired by the VectorNet proposed by Gao et al. [18], the structure of the policy network proposed in the paper is shown in Fig. 1.

The state in the environment can be seen as a series of arrays. Each array is composed of many points. For example, an array can represent the edge of a lane or a trajectory of a traffic participant. Therefore, we first consider using a graph network to model the relationship between these points in an array. The features of each array can be obtained by readout from the graph after aggregation. Then, we can get one ego feature, multiple lane features, and several traffic participant features. Combined with the type embedding representation, these features are put into a multi-head attention layer to build the interactive relationship among the ego vehicle, lanes, and other traffic participants. Then, a fully connected layer is used to output the ego's action.

Algorithm 2 Fixed-Horizon Constrained Policy Learning with Dual Ascent

Input: Initialize parameters $\boldsymbol{\theta}$ of the policy network $\pi_{\boldsymbol{\theta}}$. Initialize parameters $\boldsymbol{\psi}, \boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2$ of the three critic networks $V_{\boldsymbol{\psi}}^{\pi_{\boldsymbol{\theta}}}, \mathcal{V}_{\boldsymbol{\varphi}_1}^{\pi_{\boldsymbol{\theta}}}, \mathcal{W}_{\boldsymbol{\varphi}_2}^{\pi_{\boldsymbol{\theta}}}$, respectively. Initialize a clip ratio ϵ, a learning rate α, and two positive dual variables u, v. Initialize a data buffer $\mathcal{D}$.

Procedure:

```
1:  for each epoch do
2:      for each step do
3:          Observe state $\boldsymbol{s}$ and take the action $\boldsymbol{a} \sim \pi_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{s})$.
4:          Observe the next state $\boldsymbol{s}'$, reward $r$, safety cost $c^s$,
            comfort cost $c^c$, and done signal $d_s$.
5:          Store the data $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}', r, c^s, c^c, d_s)$ in $\mathcal{D}$.
6:      end for
7:      Compute $A^{\pi_{\boldsymbol{\theta}}}$ with $V_{\psi}^{\pi_{\boldsymbol{\theta}}}$ and data in $\mathcal{D}$.
8:      Update $\boldsymbol{\theta}$ by gradient decent using (18) and $A^{\pi_{\boldsymbol{\theta}}}$.
9:      Update $u, v$ by dual ascent using (19) and (20).
10:     Update $\boldsymbol{\psi}, \boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2$ by gradient descent with MSE loss.
11:     Clear buffer $\mathcal{D}$.
12: end for
```

Output: $\pi_{\boldsymbol{\theta}}$.

E. Policy Training With Dual Ascent Algorithm

To efficiently train the policy network, make the collision rate as small as possible (safety), and make the driving of the ego vehicle comfortable, we construct the following problem with fixed-horizon constraints:

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & -\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \\ \text{s.t.} \quad & \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\frac{1}{N}\sum_{k=0}^{N} c_{t+k}^s\right] \le C^s, \\ & \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\frac{1}{N}\sum_{k=0}^{N} c_{t+k}^c\right] \le C^c, \end{aligned} \tag{10}$$

in which N is a fixed-horizon [33], and $\boldsymbol{\theta}$ is the parameters of the policy network $\pi_{\boldsymbol{\theta}}$. The optimization goal is to maximize the discounted cumulative reward and ensure that the ratio of collisions between the ego vehicle and other traffic participants in a fixed-horizon is less than a small threshold C^s. In addition, the average number of sudden jerks in a fixed-horizon should also be less than a small threshold C^c.

As shown in Fig. 4, in existing typical constrained RL, the constraints are usually constructed with infinite-horizon cumulative cost (e.g., $\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_{k=0}^{\infty} c_{t+k}\right] \le C$) [34]. The consideration of the cost with infinite steps makes the agent conservative in the exploration process. However, with fixed-horizon constraints in this paper, the agent is not influenced by unsafe behaviors in the distant future. Thus, the constraint is actually relaxed, and it can make the agent explore with less conservativeness while keeping a sense of risk avoidance. Then, higher returns become easier to achieve.

To solve the problem shown in (10), three critic networks $V_{\boldsymbol{\psi}}^{\pi_{\boldsymbol{\theta}}}, \mathcal{V}_{\boldsymbol{\varphi}_1}^{\pi_{\boldsymbol{\theta}}}, \mathcal{W}_{\boldsymbol{\varphi}_2}^{\pi_{\boldsymbol{\theta}}}$ are used to evaluate the cumulative reward,

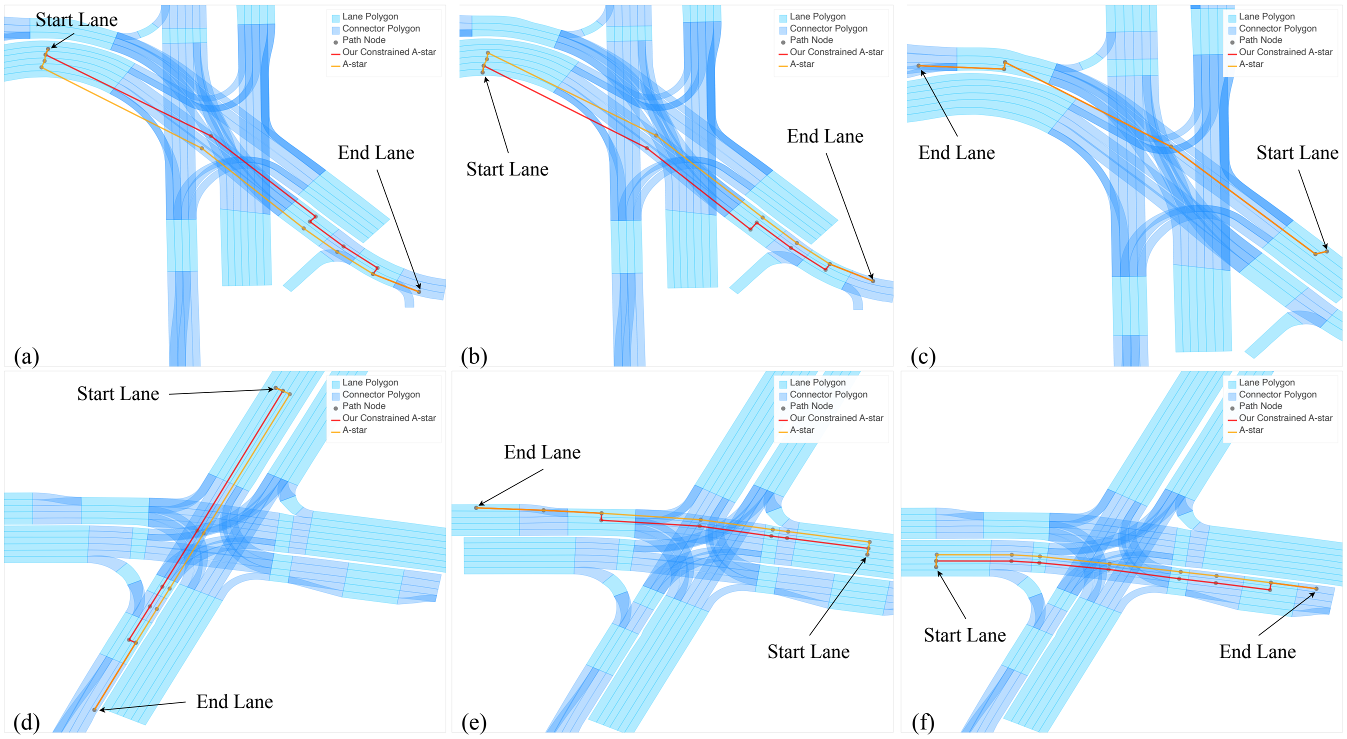


Fig. 5. Comparison of the proposed constrained A-star algorithm with the typical A-star algorithm on six different path planning tasks.

fixed-horizon safety cost, and fixed-horizon comfort cost:

$$V_{\psi}^{\pi_{\theta}}(s) = \mathbb{E}_{\pi_{\theta}}\left[\sum_{t=0}^{\infty}\gamma^t r_t \middle| s\right], \tag{11}$$

$$\mathcal{V}_{\varphi_1}^{\pi_{\theta}}(s) = \mathbb{E}_{\pi_{\theta}}\left[\frac{1}{N}\sum_{k=0}^{N} c_{t+k}^{s} \middle| s\right], \tag{12}$$

$$\mathcal{W}_{\varphi_2}^{\pi_{\theta}}(s) = \mathbb{E}_{\pi_{\theta}}\left[\frac{1}{N}\sum_{k=0}^{N} c_{t+k}^{c} \middle| s\right], \tag{13}$$

where $\psi, \varphi_1, \varphi_2$ are the parameters of the three networks $V_{\psi}^{\pi_{\theta}}, \mathcal{V}_{\varphi_1}^{\pi_{\theta}}, \mathcal{W}_{\varphi_2}^{\pi_{\theta}}$, respectively. Then, the loss functions of these three critic networks can be defined as the mean square error (MSE) on the data that has been collected by the policy. Subsequently, the advantage function $A^{\pi_{\theta}}(s_t, a_t)$ in terms of reward can be defined as

$$A^{\pi_{\theta}}(s_t, a_t) = \mathbb{E}_{\pi_{\theta}}\left[r_t + \gamma V_{\psi}^{\pi_{\theta}}(s_{t+1}) - V_{\psi}^{\pi_{\theta}}(s_t)\right], \tag{14}$$

which indicates how much is a certain action a_t a good decision given a state s_t. Practically, we can calculate the advantage using generalized advantage estimation (GAE) [35]

$$A^{\pi_{\theta}}(s_t, a_t) \approx \sum_{t'=t}^{\infty}(\gamma\lambda)^{t'-t}\delta_{t'}, \tag{15}$$

in which $\delta_{t'} = r_{t'} + \gamma V_{\psi}^{\pi_{\theta}}(s_{t'+1}) - V_{\psi}^{\pi_{\theta}}(s_{t'})$. Then, we denote $A^{\pi_{\theta}}(s_t, a_t)$ as $A_t^{\pi_{\theta_k}}$. Same as in PPO [36], to avoid unstable updates of the policy, the optimization goal in the k-th policy iteration can be set to minimize

$$L(\theta) = -\mathbb{E}_{\pi_{\theta}}\left[\min\left\{\rho_t A_t^{\pi_{\theta_k}}, \text{clip}\left(\rho_t, 1-\epsilon, 1+\epsilon\right) A_t^{\pi_{\theta_k}}\right\}\right], \tag{16}$$

where $\rho_t = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$ is the weight of the importance between π_{θ} and π_{θ_k} at time step t. clip is defined as $\text{clip}(x, b, u) = \max\{\min\{x, u\}, b\}$, and ϵ describes the clip range. Then, to solve the optimization problem with constraints shown in (10), we first denote $\mathbb{E}_{\pi_{\theta}}[1/N\sum_{k=0}^{N} c_{t+k}^{s}]$ as $J_s(\theta)$ and $\mathbb{E}_{\pi_{\theta}}[1/N\sum_{k=0}^{N} c_{t+k}^{c}]$ as $J_c(\theta)$. Then, the Lagrangian of the optimization problem in (10) can be written as

$$\mathcal{L}(\theta, u, v) = L(\theta) + u\left(J_s(\theta) - C^s\right) + v\left(J_c(\theta) - C^c\right), \tag{17}$$

where u, v are the Lagrange multipliers. With the dual ascent method, in each iteration of the policy, we can update the parameter θ with the following form:

$$\theta \leftarrow \theta - \alpha\frac{\partial\mathcal{L}}{\partial\theta}(\theta, u, v), \tag{18}$$

where α is the learning rate. Then, the Lagrange multipliers u, v can be updated by

$$\begin{aligned} u \leftarrow & \max\left\{0, u + \alpha\frac{\partial\mathcal{L}}{\partial u}(\theta, u, v)\right\}, \\ & = \max\left\{0, u + \alpha(J_s(\theta) - C^s)\right\}, \end{aligned} \tag{19}$$

$$\begin{aligned} v \leftarrow & \max\left\{0, v + \alpha\frac{\partial\mathcal{L}}{\partial v}(\theta, u, v)\right\}, \\ & = \max\left\{0, v + \alpha(J_c(\theta) - C^c)\right\}. \end{aligned} \tag{20}$$

Then, we can iteratively apply (18), (19), and (20) to update the policy and the two dual variables u, v. Finally, the problem shown in (10) can be solved, and we can obtain a policy that satisfies the constraints and has a high cumulative reward. We summarize the policy training process in Algorithm 2.

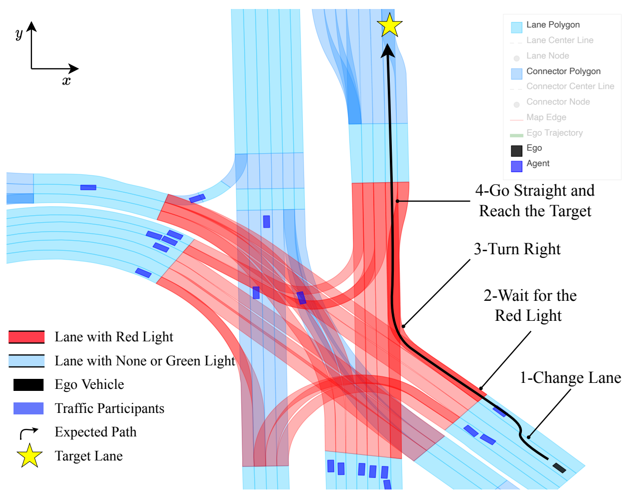


Fig. 6. The description of an autonomous driving task. During the whole process, from the initial lane, the ego vehicle needs to complete four tasks: (1) Change lanes to the right-turn lane; (2) Wait for the red light; (3) Turn right; (4) Go straight to reach the target lane. Note that the color on the lane in the intersection indicates the status of the traffic light at the current moment, and red indicates that the traffic light on this lane is red. In addition, in this environment, the time interval of the simulation is 0.1 seconds.

IV. Experiments

In this section, we first evaluate the effectiveness of the proposed constrained A-star algorithm in intersection scenarios. Then, based on the state space constructed by the constrained A-star algorithm, the performance of the proposed policy learning algorithm is evaluated in a simulation environment. Subsequently, comparisons are made with existing rule-based, IL-based, and RL-based methods to evaluate the strengths of the proposed algorithm. Overall, we aim to investigate the following research questions:

- **RQ1**: When constructing a state in the environment for incorporating the global routing information, is the proposed constrained A-star algorithm able to avoid the situation of constant lane changes during path planning?
- **RQ2**: Can the ego trained by the proposed algorithm drive to the target lane safely in autonomous driving tasks?
- **RQ3**: Compared with existing rule-based, IL-based, or RL-based approaches, can the proposed method achieve better performance?

A. Implementation Details

In this paper, we built a simulation environment based on Bokeh[2] - an interactive visualization library. This environment simulates a complex intersection whose state space, action space, and reward design have been described in Section III. Compared with other existing autonomous driving simulation environments [37–40], our environment uses real lane data (an intersection in Boston, USA), and the relationship between the lanes is more complex. In addition, traffic lights are simulated in our environment. Moreover, interactive simulations of other traffic participants were also realized by adopting IDM [8] and a lane change model [5]. During environment initialization, each traffic participant is assigned a target lane, which is chosen randomly from all nodes in the lane graph that do not have a successor node.

[2]https://bokeh.org

TABLE I
Hyperparameters Used in Our Policy Optimization Algorithm

Hyperparameter	Value
Fixed-Horizon N	10
Safety threshold C^s	0
Comfort threshold C^c	0.1
Discount factor γ	0.99
GAE λ^{GAE}	0.97
Activation function	ReLu
Learning rate of the policy network α	1e-4
Learning rate of all critic networks	2e-4
Clip ratio	0.2
Max Kullback-Leibler divergence	0.01
Max episode length	1000
Training steps of the policy network per epoch	40
Training steps of all critic networks per epoch	80
Steps per epoch	1024

One evaluation indicator is that whether the ego vehicle can successfully reach the target lane. Going out of driveable areas is considered unsuccessful. We also evaluate the number of times the ego vehicle collides with other vehicles, and the comfort level of the ego vehicle while driving (the number of jerks in an entire episode).

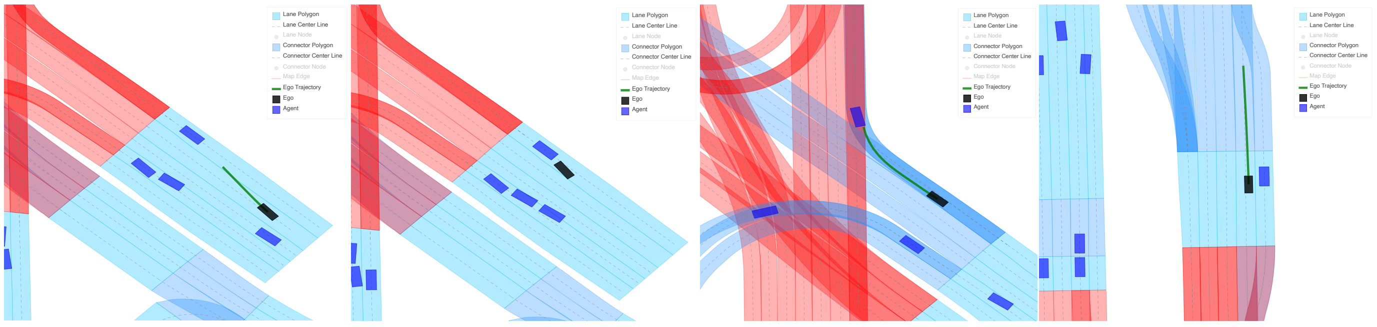
The proposed model was trained on a workstation with CPU of Intel Gold 6240R and GPU of NVIDIA GeForce RTX 3090 using PyTorch [41]. Hyperparameters of the proposed algorithm and the training configuration are shown in Table I.

B. Constrained A-star Evaluation

To answer **RQ1**, Fig. 5 shows the experimental results of the proposed constrained A-star algorithm on six path planning tasks. Compared with the traditional A-star algorithm, our method can successfully perform path planning and effectively avoid the situation of constant lane changes. Notably, in some cases, the traditional A-star method can also obtain the solution of non-constant lane changes, shown in Fig. 5 (c), but this situation cannot be guaranteed in all path planning tasks.

C. Motion Planner Evaluation

To answer **RQ2**, we evaluate the proposed policy learning algorithm in a target reach task, which is shown in Fig. 6. In this task, the ego vehicle needs to reach the target lane safely, which means the ego vehicle should not collide with other traffic participants, go out of the drivable area, or violate traffic rules. As shown in Fig. 6, starting from the initial lane, firstly, the ego should change to the right-turn lane. Then, the agent may need to wait for the end of the red light at the intersection. Subsequently, the ego needs to turn right and finally go straight to the target lane. During this period, in order to avoid collisions with other vehicles, the ego may also need to follow the car ahead. To train the policy network by the proposed constrained RL algorithm, the simulation environment is built based on the map shown in Fig. 6. Fig. 7 shows the key trajectories of the learned policy in



(a) Change lanes (b) Wait for the end of the red light (c) Turn right (d) Go straight

Fig. 7. Trajectories of the policy trained by the proposed method in our simulation environment with four subtasks, including changing lanes, waiting for the red light, turning right, and going straight. The green lines are the trajectories of the ego vehicle in the next 1.5 seconds. A more detailed driving video can be viewed in the supplementary materials.

performing the autonomous driving task. It can be seen that the ego vehicle can successfully complete tasks such as changing lanes, waiting for the red light, and turning right. Finally, the policy obtained by the proposed method successfully and safely arrives at the target lane. In addition, the position, heading angle, velocity and angular velocity of the ego vehicle when driving to the target position is shown in Fig. 8. The whole driving process took about 26 seconds, and the period is divided into 4 stages, namely changing lanes, waiting for a red light, turning right and going straight to approach the target point. Among them, the maximum velocity is 19.903 m/s, with an average speed of 7.518 m/s.

Therefore, we can conclude that the driving policy obtained by the proposed algorithm can successfully make the ego vehicle drive to the target position safely.

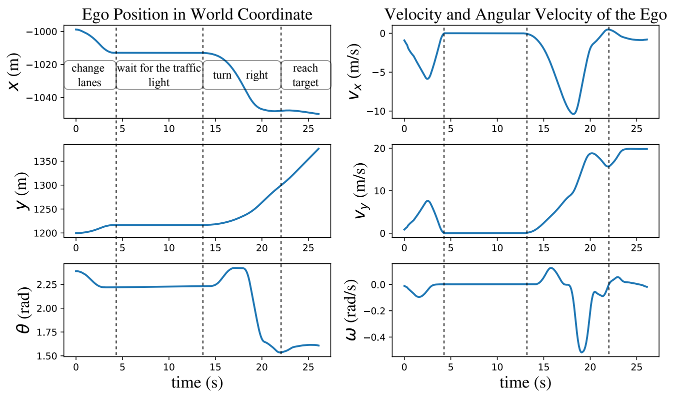


Fig. 8. The position and velocity of the ego vehicle during performing an autonomous driving task, in which the ego's motion planning policy is trained by the proposed policy optimization algorithm.

D. Compare With Related Methods

To answer **RQ3**, we compare the proposed policy learning algorithm with a traditional rule-based planning approach, two IL-based approaches, and one RL-based method. Specifically, the following four methods are compared:

- A rule-based method: this method is a decision-making and planning model, based on the IDM [8] model and a lane-changing model [5]. In order to further ensure the safety of the model, an additional emergency braking rule is added to the model. When the ego vehicle is less than 1 meter away from the vehicle in front, the ego vehicle will brake urgently to avoid a collision.
- An IL-based method (IL-1): a common IL-based planning method is Urban Driver [19]. The model is trained on the Lyft dataset [42], which includes 100 hours of expert demonstrations on urban roads.
- An IL-based method (IL-2): considering the fact that a larger scale autonomous driving planning dataset has been released recently [40], named NuPlan Dataset[3], which contains approximately 1300 hours of driving data with a total size of about 1.8T, we fine-tune the Urban Driver model (IL-1) on these data and get a new IL-based model (IL-2).
- A typical deep RL method: PPO [36] is a classic deep RL algorithm, which is featured by gradient clipping. In experiments, PPO is also trained in the proposed environment. Without the constrained A-star, the representation of lanes is slightly different. The way to get the lane state is as follows: (1) Given the roads where the starting point and the end point are located, the traditional A-star algorithm is used to obtain the navigation roads where the agent should drive. Note that a road is composed of multiple side-by-side lanes in the same direction; (2) At time step t, lane state is all the lanes on the navigation road near the ego car. Without the constrained A-star, no lane-level standard trajectory τ_t can be used to reward the agent. Instead, the centerline of the road is used as the reference trajectory. In addition, the policy network used in PPO is the same as the policy network presented in this paper.

[3]https://www.nuscenes.org/nuplan#data-collection

Table II shows the evaluation results of the above four approaches and the proposed policy learning algorithm in the simulation environment with the indicators of total reward, total safety cost, and total comfort cost. It can be seen from Table II that the rule-based method can successfully

TABLE II
COMPARISON OF OUR DRIVING POLICY WITH THE RELATED METHODS

Method	Performance			Reach Target	Time and Speed		
	Total Reward	Total Safety Cost (Number of Collisions)	Total Comfort Cost (Number of Jerks)		Time Consumed	Average Velocity	Maximum Velocity
Rule-based [5, 8]	246.526	0	24	Yes	28.9 s	6.779 m/s	14.258 m/s
IL-1 [19]	253.316	3	12	Yes	27.6 s	7.094 m/s	18.283 m/s
IL-2 [19, 40]	249.464	2	9	Yes	28.7 s	6.860 m/s	17.946 m/s
PPO [36]	56.697	6	16	No	N/A	N/A	N/A
Ours	262.308	0	18	Yes	26.2 s	7.518 m/s	19.903 m/s

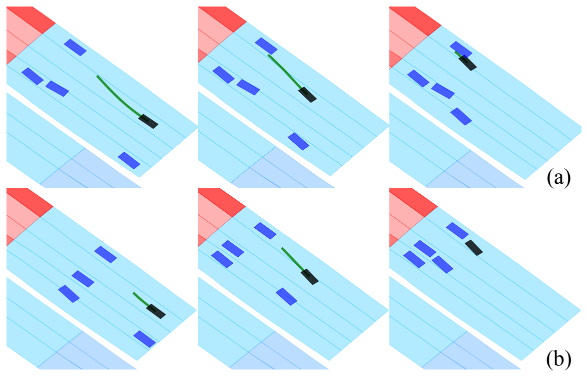


Fig. 9. Visual comparisons of avoiding collisions between our method and the IL-2 method. (a) Visualization result of IL-2 method; (b) Visualization result of the proposed method.

TABLE III
ABLATION STUDY OF THE PROPOSED POLICY LEARNING ALGORITHM

Model	Total Reward	Total Safety Cost	Total Comfort Cost
WCC*	274.941	0	33
WSC†	277.215	21	17
IH‡	229.692	0	11
Ours	262.308	0	18

* WCC means without the comfort cost constraint.
† WSC means without the safety cost constraint.
‡ IH means infinite-horizon in constraints.

reach the target lane and has not collided with other traffic participants. However, due to the rule of emergency braking in order to ensure safety, there are 24 jerks during driving. Moreover, the total reward attained by the rule-based method is relatively low. The number of jerks in IL-based methods is significantly reduced due to training from a large number of expert demonstrations. However, because the training goals of IL-based are relatively single and cannot effectively weigh between safety and goal completion, it leads to a certain collision. The driving speed of IL-based methods is higher than the rule-based method, and it can get higher total rewards. For PPO, it can't reach the target lane. This is mainly because the trajectory is quite aggressive without considering the comfort constraint and safety constraint during policy learning, which eventually leads to deviation from the drivable area especially when the ego vehicle turns. Moreover, the lack of lane-level representation and reward design may also be one of the reasons. As for the proposed constrained policy learning algorithm, the highest total reward is achieved. The proposed method also achieves zero collision due to an additional safety constraint term during policy learning. Furthermore, from the aspect of comfort (number of jerks), the method is superior to the rule-based method but inferior to IL-based methods. Moreover, the proposed method also achieves the fastest average speed and minimum time consumption.

In addition, we also visualize the performance of our method and the IL-2 method in collision avoidance, as shown in Fig. 9. Group (a) shows a situation where the IL-2 method cannot avoid a collision with a car in front. The ego car has a tendency to brake, but it does not stop completely, which means the planning network does not output 0 absolutely in such a case. In similar cases, our method is able to stop stably behind the car in front, as shown in group (b) in Fig. 9. This is due to the safety constraint in the proposed method, which guarantees that the final learned planner can avoid collisions.

From these observations, we can obtain the conclusion that the proposed constrained policy learning method can achieve a better balance in terms of reaching target, safety, and comfort in the simulated autonomous driving task.

E. Ablation Study

To further verify the effects of various components in the proposed motion planner, we perform an ablation experiment on the proposed method on the components of comfort cost constraint, safety cost constraint, and the horizon in constraints. Table III shows the result of the ablation study. It can be observed that removing constraints yields a higher total return, but the corresponding safety or comfort will be reduced. In addition, increasing the horizon in the constraint (e.g., infinite horizon) helps achieve better safety and comfort, but will also significantly affect the value of the total reward. As a consequence, with appropriate constraints, the final policy network can achieve a good balance in goal completion, safety, and comfort.

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed a fixed-horizon constrained RL framework for decision-making and trajectory planning problems in autonomous driving, which can make autonomous vehicles reach the target point safely and comfortably in complex transportation environments. Different from many existing IL-based methods, which are aimed at short-term trajectory planning, this paper introduces global navigation information in the state representation for long-term planning. To avoid constant lane changes in path planning for global routing, we first propose the constraint A-star algorithm, in which the optimality is guaranteed theoretically that the optimal path under the constraint of non-constant lane changes can be reached. Therefore, as a sort of global navigation information, the result of constrained path planning is utilized in the lane representation. Then, to achieve a good balance among reaching target goals, safety, and comfort in the task of planning, we proposed the fixed-horizon constrained policy learning algorithm, in which safety and comfort are constructed as two different constraints. The goal of the agent is to reach the target point while satisfying the constraints. The dual ascent technique is proposed to solve the constrained RL problem. In addition, to reduce the conservativeness of the agent in the learning phase, we construct the safety constraint and comfort constraint as the paradigm of fixed-horizon cumulative cost. Complex traffic simulation experiments show that the trajectory planning policy obtained by the proposed algorithm can successfully perform the tasks of changing lanes, waiting for traffic lights, turning, and going straight. Things like collisions with other vehicles and violations of traffic lights do not happen during driving. Compared with other methods, the proposed method can get a better total reward and can better balance safety, comfort, and reaching targets.

It is worth noting that the trajectories given by the proposed method are not as comfortable as the methods based on IL. The main reason is that IL-based methods are trained on expert demonstration data so that the trajectory output by the network is similar to the trajectory driven by a human, which might be smoother. However, our method discretely gives the waypoint after 0.1 seconds to the agent, so the overall comfort is not as good as methods based on IL. In future work, we will deal with this problem by combining RL and expert demonstration. In addition, we also consider applying the proposed algorithm to real scenarios.

References

[1] Z. Zhu and H. Zhao, "A survey of deep RL and IL for autonomous driving policy learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 14 043–14 065, 2022.

[2] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, 2022.

[3] Y. Huang, J. Du, Z. Yang, Z. Zhou, L. Zhang, and H. Chen, "A survey on trajectory-prediction methods for autonomous driving," *IEEE Trans. Intell. Vehicles*, vol. 7, no. 3, pp. 652–674, 2022.

[4] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 740–759, 2022.

[5] J. Erdmann, "SUMO's lane-changing model," in *Proc. Model. Mobility Open Data: 2nd Sumo Conf.* Springer, 2015, pp. 105–123.

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

[7] V. A. Butakov and P. Ioannou, "Personalized driver/vehicle lane change models for ADAS," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4422–4431, 2015.

[8] M. Treiber and A. Kesting, *Traffic Flow Dynamics: Data, Models and Simulation*. Springer, 2013.

[9] M.-J. Wierbos, V. Knoop, F. Hänseler, and S. Hoogendoorn, "A macroscopic flow model for mixed bicycle–car traffic," *Transportmetrica A: Transport Sci.*, vol. 17, no. 3, pp. 340–355, 2021.

[10] A. Jafaripournimchahi, Y. Cai, H. Wang, L. Sun, Y. Tang, and A. A. Babadi, "A viscous continuum traffic flow model based on the cooperative car-following behaviour of connected and autonomous vehicles," *IET Intell. Transport Syst.*, early access, 2022.

[11] A. Kurt and Ü. Özgüner, "Hierarchical finite state machines for autonomous mobile systems," *Control Eng. Pract.*, vol. 21, no. 2, pp. 184–194, 2013.

[12] J.-F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal, *Numerical optimization: theoretical and practical aspects*. Springer Science & Business Media, 2006.

[13] B.-C. Seet, G. Liu, B.-S. Lee, C.-H. Foh, K.-J. Wong, and K.-K. Lee, "A-star: A mobile ad hoc routing strategy for metropolis vehicular communications," in *Proc. Int. Conf. Res. Netw.* Springer, 2004, pp. 989–999.

[14] X. Wang, K. Tang, X. Dai, J. Xu, J. Xi, R. Ai, Y. Wang, W. Gu, and C. Sun, "Safety-balanced driving-style aware trajectory planning in intersection scenarios with uncertain environment," *IEEE Trans. Intell. Vehicles*, early access, 2023.

[15] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," in *Proc. Robotics: Sci. Syst.*, 2019.

[16] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun, "End-to-end interpretable neural motion planner," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 8660–8669.

[17] S. Casas, A. Sadat, and R. Urtasun, "MP3: A unified model to map, perceive, predict and plan," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, 2021, pp. 14 403–14 412.

[18] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding HD maps and agent dynamics from vectorized representation," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, 2020, pp. 11 525–11 533.

[19] O. Scheel, L. Bergamini, M. Wolczyk, B. Osiński, and P. Ondruska, "Urban driver: Learning to drive from real-world demonstrations using policy gradients," in *Proc. Conf. Robot Learn.*, 2021.

[20] D. Chen and P. Krähenbühl, "Learning from all vehicles," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, 2022, pp. 17 222–17 231.

[21] E. Bronstein, M. Palatucci, D. Notz, B. White, A. Kuefler, Y. Lu, S. Paul, P. Nikdel, P. Mougin, H. Chen, J. Fu, A. Abrams, P. Shah, E. Racah, B. Frenkel, S. Whiteson, and D. Anguelov, "Hierarchical model-based imitation learning for planning in autonomous driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2022, pp. 8652–8659.

[22] G. C. K. Couto and E. A. Antonelo, "Generative adversarial imitation learning for end-to-end autonomous driving on urban environments," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*. IEEE, 2021.

[23] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, 2021.

[24] Z. Huang, J. Wu, and C. Lv, "Efficient deep reinforcement learning with imitative expert priors for autonomous driving," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, 2022.

[25] Y. Zhao, K. Wu, Z. Xu, Z. Che, Q. Lu, J. Tang, and C. H. Liu, "CADRE: A cascade deep reinforcement learning framework for vision-based autonomous urban driving," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, no. 3, Jun. 2022, pp. 3481–3489.
[26] B. Gangopadhyay, P. Dasgupta, and S. Dey, "Safe and stable RL (S2RL) driving policies using control barrier and control Lyapunov functions," *IEEE Trans. Intell. Vehicles*, early access, 2022.
[27] Y. Hu, J. Fu, and G. Wen, "Safe reinforcement learning for model-reference trajectory tracking of uncertain autonomous vehicles with model-based acceleration," *IEEE Trans. Intell. Vehicles*, early access, 2023.
[28] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, 2022.
[29] C. Bai, T. Xiao, Z. Zhu, L. Wang, F. Zhou, A. Garg, B. He, P. Liu, and Z. Wang, "Monotonic quantile network for worst-case offline reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, 2022.
[30] K. Lin, D. Li, Y. Li, S. Chen, Q. Liu, J. Gao, Y. Jin, and L. Gong, "TAG: Teacher-advice mechanism with Gaussian process for reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, 2023.
[31] E. Altman, *Constrained Markov decision processes*. Routledge, 1999.
[32] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[33] K. De Asis, A. Chan, S. Pitis, R. Sutton, and D. Graves, "Fixed-horizon temporal difference methods for stable reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 04, 2020, pp. 3741–3748.
[34] Y. Liu, A. Halev, and X. Liu, "Policy learning with constraints in model-free reinforcement learning: A survey," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, 2021.
[35] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv:1506.02438*, 2015.
[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.
[37] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. Conf. Robot Learn.* PMLR, 2017.
[38] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo–simulation of urban mobility: An overview," in *Proc. Third Int. Conf. Advances System Simul.* ThinkMind, 2011.
[39] Q. Li, Z. Peng, L. Feng, Q. Zhang, Z. Xue, and B. Zhou, "Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning," *IEEE Trans. Pattern Anal. and Mach. Intell.*, early access, 2022.
[40] H. Caesar, J. Kabzan, K. S. Tan, W. K. Fong, E. Wolff, A. Lang, L. Fletcher, O. Beijbom, and S. Omari, "Nuplan: A closed-loop ML-based planning benchmark for autonomous vehicles," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit. ADP3 workshop*, 2021.
[41] A. Paszke, S. Gross *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.
[42] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, L. Chen, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, "One thousand and one hours: Self-driving motion prediction dataset," in *Proc. Conf. Robot Learn.* PMLR, 2021, pp. 409–418.

Ke Lin (Graduate Student Member, IEEE) received the B.E. degree in mechanical engineering from the China University of Geosciences, Wuhan, China, in 2017, and the M.E. degree in mechanical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2020. He is currently working toward the Ph.D. degree in control science and engineering with the Harbin Institute of Technology (Shenzhen), Shenzhen, China. His research interests include safe reinforcement learning and its applications in autonomous driving.

Yanjie Li (Member, IEEE) received the B.S. degree from Qingdao University (QDU), Qingdao, China, in 2001 and Ph.D. degree from the University of Science and Technology of China (USTC), Hefei, China, in 2006. From August 2006 to August 2008, he was a Research Associate in the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology (HKUST). In September 2008, he joined Harbin Institute of Technology Shenzhen (HITSZ), Shenzhen, China. Now he is an associate professor at HITSZ. He is the recipient of Ho-Pan-Ching-Yi best paper award in 2014. His research interests include stochastic optimization and learning, Markov decision process (MDP), partially observable MDP and reinforcement learning.

Shiyu Chen received the B.E. degree in automation from Anyang Institute of Technology, Anyang, China, in 2014, and M.E. degree in control science and engineering from Harbin Institute of Technology Shenzhen (HITSZ), Shenzhen, China, in 2017. He is currently working toward the Ph.D. degree in control science and engineering at HITSZ, Shenzhen, China. His research interests include deep reinforcement learning, and agile trajectory planning.

Duantengchuan Li received the M.E. degrees from the School of Computer Science and Technology, Central China Normal University, Wuhan, China, in 2021. He is currently working toward the Ph.D. degree in software engineering with the School of Computer Science, Wuhan University, Wuhan, China. His research interests include deep learning, and reinforcement learning.

Xinyu Wu (Senior Member, IEEE) received the B.S. and M.E. degrees from the Department of Automation, University of Science and Technology of China (USTC), Hefei, China, in 2001 and 2004, respectively, and the Ph.D. degree from The Chinese University of Hong Kong (CUHK), China, in 2008. He is currently a Professor with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China, where he is the Director of the Center for Intelligent Bionic. He has published over 180 papers and two monographs. His research interests include computer vision, robotics, and intelligent system. Prof. Wu is an Associate Editor of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, and IEEE ROBOTICS AND AUTOMATION LETTERS.