ICCAD 2020: International Conference on Connected Automated Driving

Lane-change Path Planning of Autonomous Driving using Model-based Optimization, Deep Reinforcement Learning and 5G Vehicle-to-vehicle (V2V) Communications

William Li,
Delbarton School, Morristown, NJ 07960, USA
E-mail: `li_w@delbarton.org`

***Abstract*—Lane-change path planning is a crucial and yet complex task in autonomous driving. The traditional path planning approach based on a system of carefully-crafted rules to cover various driving scenarios becomes unwieldy as more and more rules are added to deal with exceptions and corner cases. This paper proposes to divide the entire path planning to two stages. In the first stage the ego vehicle travels longitudinally in the source lane to reach a safe state. In the second stage the ego vehicle makes lateral lane-change maneuver to the target lane. The paper derives the safe state conditions based on lateral lane-change maneuver calculation to ensure collision free in the second stage. To determine the acceleration sequence that minimizes the time to reach a safe state in the first stage, the paper proposes three schemes, namely, kinetic model based optimization, deep reinforcement learning, and 5G vehicle-to-vehicle (V2V) communications. The paper investigates these schemes via simulation. The model-based optimization is sensitive to the model assumptions. The deep reinforcement learning is more flexible in handling scenarios beyond the model assumed by the optimization. The 5G V2V eliminates uncertainty in predicting future behaviors of surrounding vehicles by sharing driving intents and enabling cooperative driving.**

***Index Terms*—Lane change, path planning, autonomous driving, deep reinforcement learning, 5G, V2V communications, connected vehicles.**

I. Introduction

Autonomous driving requires a great level of intelligence, which can roughly be divided into three categories [1]: *recognition* that identifies components of the surrounding environment, *prediction* that predicts the future states of the environment, and *planning* that incorporates recognition and prediction to plan the future sequence of driving actions. Planning is the hardest task of the three, because it requires decision making in complex driving environments. One of planning tasks is lane-change. Lane-change is a highly demanding task because the autonomous vehicle needs to watch the leading vehicle on its source lane and surrounding vehicles on the target lane and to optimize its maneuver to safely arrive at its destination. According to one study [2], around 10% of all freeway crashes are caused by lane-change. Therefore, well-designed automated lane-change can significantly enhance the driving safety.

Lane-change involves both *path planning* that plans the trajectory of the ego vehicle and *drive-by-wire control* that uses electrical or electro-mechanical systems to move the vehicle following the planned trajectory. This paper focuses on the path planning part of lane-change.

The traditional path planning approach is based on a system of carefully-crafted rules to cover various driving scenarios [3]. A major drawback is that the system may become unwieldy as more and more rules are added to deal with exceptions and corner cases. While individual rules may be easy to interpret, the complex interactions of many rules is usually difficult to grasp. Recently a number of neural networks have been presented in the literature for lane-change path planning [4]–[7], where the goal is to obtain a safe and smooth driving policy through deep reinforcement learning. These algorithms are studied via simulation to numerically demonstrate that they can effectively achieve the goal. However, the performance of the algorithms has not been benchmarked. For example, it is unclear whether the planned path is optimal and efficient.

A novelty of this paper is that the paper first derives an optimal lane-change path planning algorithm based on a kinetic speed control model, and then compares its performance with that using deep reinforcement learning. The results show that the deep reinforcement learning scheme can outperform the model-based optimization scheme when the travel pattern is beyond the model assumed by the latter. Thus, deep learning is more flexible in handling different scenarios and more robust overall with sufficient learning.

In the above model-based optimization and deep learning schemes, a vehicle makes its own path planning decision without explicitly exchanging information with other vehicles. A key challenge is thus the prediction of future motions of surrounding vehicles. 5G vehicle-to-vehicle (V2V) communications enables vehicles to talk to each other with very low latency and high reliability.

Another novelty of this paper is that the paper studies the performance improvement resulted from the support of 5G V2V. In particular, 5G V2V allows a vehicle to be informed of the future path trajectories of surrounding vehicles and to optimize its path planning accordingly (passive cooperation). Furthermore, the vehicle can inform surrounding vehicles of its intent of lane-change and request them to adjust their own path trajectories to facilitate lane-change of the vehicle (active cooperation).

The remainder of the paper is organized as follows. Section II describes the system model assumption in this paper.

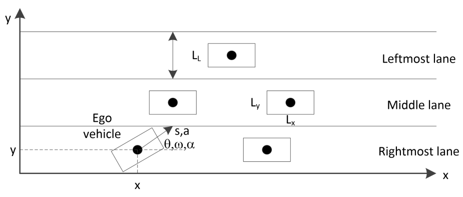


Fig. 1: Illustration of the vehicle and road model.

Section III proposes the two-stage path planning, defines the notion of safe state and derives its condition. Section IV proposes a model-based optimization algorithm where the model assumes surrounding vehicles travel at zero acceleration. The motions of surrounding vehicles are however more sophisticated in the real world. To address this challenge, Section V proposes a deep reinforcement learning scheme to learn from the vehicle motion data simulated with the famous Intelligent Driver Model (IDM). Section VI introduces 5G V2V and proposes the ideas of passive and active cooperation for lane-change path planning. Section VII presents the simulation results of the performance of the three proposed schemes. Finally, Section VIII concludes this paper and proposes a few areas for future research.

II. System Model

A. Vehicle and Road Model

Figure 1 illustrates the vehicle and road model. A vehicle is modeled as a 2D rectangle with dimension $L_x \times L_y$, where L_x represents the length and L_y the width. The state of the vehicle at time t is defined by $(x(t), y(t), s(t), \theta(t), \omega(t))$, where $x(t), y(t)$ are the positions of the center of the rectangle along the longitudinal and lateral axes, respectively, $s(t)$ is the speed, $\theta(t), \omega(t)$ are the yaw rotation and yaw rate, respectively.

$$\dot{x} = s\cos\theta, \dot{y} = s\sin\theta, \dot{\theta} = \omega. \tag{1}$$

Denote $a(t), \alpha(t)$ the linear and angular accelerations respectively.

$$\dot{s} = a, \dot{\omega} = \alpha. \tag{2}$$

Given the initial condition, the trajectory of the vehicle is fully controlled by the accelerations $a(t), \alpha(t)$ determined by path planning. For safe driving, assume the maximum and minimum speeds $s_{\min} \le s(t) \le s_{\max}$. $|a(t)| \le a_{\max}$ where $a_{\max}$ is a parameter depending on the comfortable level for a human.

Vehicles are randomly dropped on a straight multi-lane road with lane width L_L where two vehicles in the same lane are separated by a safe distance defined in Section II-B. They all travel in the positive x-axis direction. The ego vehicle starts from the rightmost lane and intends to move to the middle lane. The goal of path planning is to complete lane-change in the shortest time without collision.

B. Kinetic Model of Longitudinal Travel

Consider longitudinal travel without lane-change. Thus, θ, ω, α are all equal to 0. Denote $s_e(t_2), s_l(t_2)$ the initial

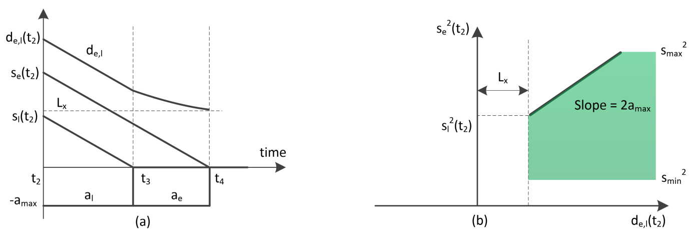


Fig. 2: Illustration of the safe distance in the worst case scenario when $s_e(t_2) > s_l(t_2)$. (a) Calculation of speeds and distance. (b) Safe distance region of $s_e^2(t_2)$ versus $d_{e,l}(t_2)$ given $s_l^2(t_2)$ as specified by (4).

speeds at some time t_2 of the ego vehicle and the *leading vehicle* that is just ahead of the ego vehicle in the same lane and $d_{e,l}(t_2) > 0$ the initial distance between the two. Denote $a_e(t), a_l(t)$ the accelerations for $t \ge t_2$. $d_{e,l}(t_2)$ is a *safe distance* if the ego vehicle can employ appropriate $a_e(t)$ to avoid hitting the leading vehicle for any $a_l(t)$. Collision occurs if $|d_{e,l}(t)| < L_x$ at any $t \ge t_2$.

If $s_e(t_2) \le s_l(t_2)$, then the safe distance satisfies $d_{e,l}(t_2) \ge L_x$. The ego vehicle can set $a_e(t) = a_l(t)$ for $t \ge t_2$, which results in $s_e(t) \le s_l(t)$ and therefore $d_{e,l}(t) \ge d_{e,l}(t_2) \ge L_x$.

If $s_e(t_2) > s_l(t_2)$, then the worst case scenario[1] for potential collision is when $a_l(t) = -a_{\max}$ until $s_l(t) = 0$ at $t = t_3$, where $t_3 - t_2 = \frac{s_l(t_2)}{a_{\max}}$, and $a_l(t) = 0$ thereafter. To avoid collision, the ego vehicle can set $a_e(t) = -a_{\max}$ until $s_e(t) = 0$ at $t = t_4$, where $t_4 - t_2 = \frac{s_e(t_2)}{a_{\max}}$, and $a_e(t) = 0$ thereafter.

Figure 2(a) shows the safe distance calculation. For $t \in [t_2, t_3]$, $s_e(t), s_l(t)$ both decrease at the rate of $-a_{\max}$, and $d_{e,l}(t) = d_{e,l}(t_2) - (s_e(t_2) - s_l(t_2))(t - t_2)$. Thus,

$$d_{e,l}(t_3) = d_{e,l}(t_2) - \frac{(s_e(t_2) - s_l(t_2))s_l(t_2)}{a_{\max}}. \tag{3}$$

For $t \in [t_3, t_4]$, $s_e(t)$ further decreases at the rate of $-a_{\max}$ while $s_l(t) = 0$, and $d_{e,l}(t) = d_{e,l}(t_3) - (s_e(t_2) - s_l(t_2))(t - t_3) + \frac{1}{2}a_{\max}(t - t_3)^2$. Thus,

$$\begin{aligned} d_{e,l}(t_4) &= d_{e,l}(t_3) - (s_e(t_2) - s_l(t_2))\frac{s_e(t_2) - s_l(t_2)}{a_{\max}} \\ &\quad + \frac{1}{2}a_{\max}\left(\frac{s_e(t_2) - s_l(t_2)}{a_{\max}}\right)^2 \\ &= d_{e,l}(t_2) - \frac{s_e^2(t_2) - s_l^2(t_2)}{2a_{\max}}. \end{aligned}$$

Hence, the safe distance is given by

$$d_{e,l}(t_2) \ge \max\left(0, \frac{s_e^2(t_2) - s_l^2(t_2)}{2a_{\max}}\right) + L_x. \tag{4}$$

Figure 2(b) shows the safe distance region derived from (4).

[1]For safety, assume here that collision must be avoided even if the leading vehicle violates the speed limits. That is, $s_l(t)$ can be as low as 0, which is below $s_{\min}$. In an emergency scenario, the leading vehicle stops completely rather than travels at the minimum $s_{\min}$.

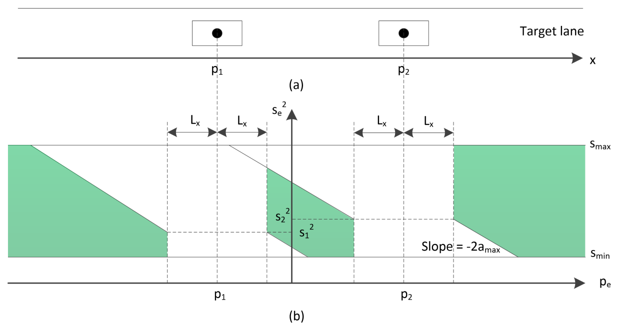


Fig. 3: Illustration of the feasible region of p_e, s_e, shown as the green areas in (b), in the presence of vehicles $1, 2$ in the target lane shown in (a). The feasible region is a function of p_1, s_1, p_2, s_2 as well as parameters $L_x, s_{\max}, s_{\min}$. The slope of the slanting lines is $-2a_{\max}$.

For the purpose of lane-change path planning, suppose that vehicles i are already in the target lane with positions p_i and speeds s_i, where $i = 1, 2, \ldots$. The ego vehicle has to find a position p_e and speed s_e with which it can insert itself to the target lane so that it keeps a safe distance with any vehicle i. Figure 3 illustrates the feasible region of p_e, s_e derived from (4). Note that keeping a safe distance with any existing vehicle after the ego vehicle has merged to the target lane is just one condition for collision-free lane-change. Section III will discuss other conditions using the kinetic model of lane-change maneuver presented next.

C. Kinetic Model of Lane-change Maneuver

To make a lane-change maneuver, the ego vehicle applies nonzero angular acceleration α. Denote τ the duration of the maneuver, which starts at some t_1. For simplicity, assume no linear acceleration (i.e., $a = 0$) during the maneuver and the following piecewise constant angular acceleration control function

$$\alpha(t) = \begin{cases} \alpha_0, & t \in [t_1, t_1 + \frac{\tau}{4}). \\ -\alpha_0, & t \in [t_1 + \frac{\tau}{4}, t_1 + \frac{3\tau}{4}). \\ \alpha_0, & t \in [t_1 + \frac{3\tau}{4}, t_1 + \tau). \end{cases} \tag{5}$$

α_0 is a parameter to be determined later in this section. θ, ω can be determined by integration of (1) and (2).

$$\omega(t) = \begin{cases} \alpha_0(t - t_1), & t \in [t_1, t_1 + \frac{\tau}{4}). \\ \alpha_0\frac{\tau}{4} - \alpha_0(t - t_1 - \frac{\tau}{4}), & t \in [t_1 + \frac{\tau}{4}, t_1 + \frac{3\tau}{4}). \\ -\alpha_0\frac{\tau}{4} + \alpha_0(t - t_1 - \frac{3\tau}{4}), & t \in [t_1 + \frac{3\tau}{4}, t_1 + \tau). \end{cases} \tag{6}$$

$$\theta(t) = \begin{cases} \frac{1}{2}\alpha_0(t - t_1)^2, \\ \frac{1}{2}\alpha_0(\frac{\tau}{4})^2 + \alpha_0\frac{\tau}{4}(t - t_1 - \frac{\tau}{4}) - \frac{1}{2}\alpha_0(t - t_1 - \frac{\tau}{4})^2, \\ \frac{1}{2}\alpha_0(\frac{\tau}{4})^2 - \alpha_0\frac{\tau}{4}(t - t_1 - \frac{3\tau}{4}) + \frac{1}{2}\alpha_0(t - t_1 - \frac{3\tau}{4})^2. \end{cases} \tag{7}$$

Figure 4 plots $\alpha(t), \omega(t), \theta(t)$, and shows that $\theta(t)$ is a smooth function of t and ω, θ both return to 0 at the end of the lane-change maneuver $t_1 + \tau$.

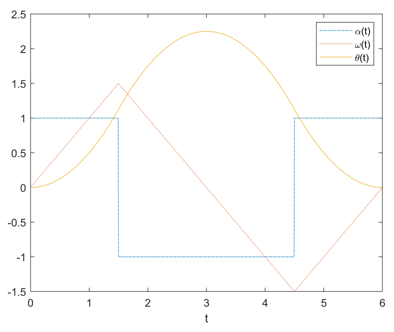


Fig. 4: Plots of $\alpha(t), \omega(t), \theta(t)$ with $\alpha_0 = 1, \tau = 6$.

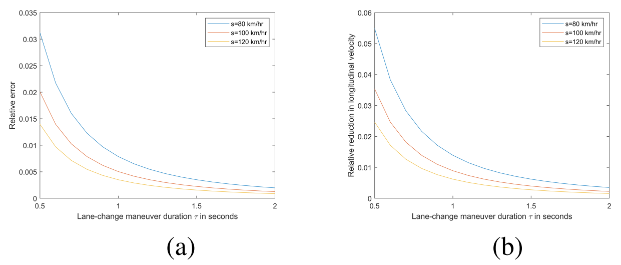


Fig. 5: Performance study of Lane-change Maneuver. (a) Relative error in the total lateral displacement $|\Delta_y - L_L|/L_L$ as a function of τ when the closed-form expression (10) is used as an approximation of α_0. (b) Relative reduction in the average longitudinal velocity from the speed $1 - \frac{\bar{v}_x}{s}$ as a function of τ. In the figure, $L_L = 3$ m.

From (7), one can calculate the lateral velocity $\dot{y}$. The total lateral displacement during the maneuver can be calculated by integrating (1),

$$\Delta_y = \int_{t_1}^{t_1+\tau} \dot{y}(t)\, dt = \int_{t_1}^{t_1+\tau} s \sin\theta(t)\, dt, \tag{8}$$

where s is a constant linear speed during the maneuver. Given τ, α_0 can be determined from $\Delta_y = L_L$. When $\theta(t) \ll 1$, $\sin\theta(t) \approx \theta(t)$, and one can obtain a closed-form expression of α_0 as follows.

$$s\alpha_0 \left(\int_0^{\frac{\tau}{4}} \frac{1}{2}t^2\, dt + \int_0^{\frac{\tau}{2}} \frac{1}{2}\left(\frac{\tau}{4}\right)^2 + \frac{\tau}{4}t - \frac{1}{2}t^2\, dt \right.$$
$$\left. + \int_0^{\frac{\tau}{4}} \frac{1}{2}\left(\frac{\tau}{4}\right)^2 - \frac{\tau}{4}t + \frac{1}{2}t^2\, dt \right) \approx L_L. \tag{9}$$

The expression in the parenthesis can be simplified to $\frac{\tau^3}{32}$. Therefore,

$$\alpha_0 \approx \frac{32 L_L}{s\tau^3}. \tag{10}$$

To assess the accuracy of the above approximation, define the relative error in the total lateral displacement as $|\Delta_y - L_L|/L_L$, where Δ_y is given in (8) and α_0 is determined from the closed-form expression (10). Figure 5(a) plots the relative

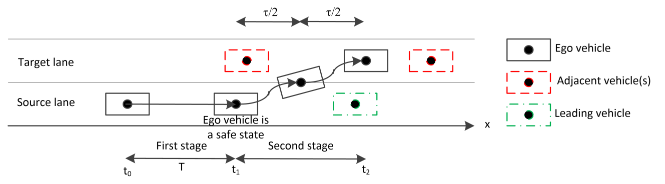


Fig. 6: Illustration of two-stage path planning of lane-change.

error versus τ. The relative error is below 1% unless τ is very small.

During the lane-change maneuver, the longitudinal velocity given in (1), $\dot{x}$, is strictly smaller than speed s. The average longitudinal velocity can be calculated by dividing the total longitudinal displacement over the maneuver duration,

$$\bar{v}_x = \frac{\Delta_x}{\tau} = \frac{\int_{t_1}^{t_1+\tau} \dot{x}(t)\,dt}{\tau} = \frac{\int_{t_1}^{t_1+\tau} s\cos\theta(t)\,dt}{\tau} < s. \quad (11)$$

The loss of the average longitudinal velocity from the speed can be measured by the relative reduction $1 - \frac{\bar{v}_x}{s}$. Figure 5(b) plots the relative reduction versus τ. The relative reduction is below 2% unless τ is very small. Therefore, in the following, to simplify the calculation, the paper ignores the difference and assumes the longitudinal velocity is equal to the speed during the lane-change maneuver.

III. Two-stage Path Planning of Lane-change

The entire path planning of lane-change is a sequence of complex operations. To address the complexity issues, this study proposes a two-stage approach illustrated in Figure 6. In the first stage, the ego vehicle travels longitudinally in the source lane with $\alpha = 0$ as in Section II-B. The goal is to reach a safe state from which the ego vehicle can proceed to the second stage. If the ego vehicle is already in a safe state, then the first stage is skipped. In the second stage, from a safe state the ego vehicle makes a lane-change maneuver to the target lane with $a = 0$ as in Section II-C. Denote T, τ the durations of the two stages, respectively. $[t_0, t_1), [t_1, t_2)$ are the respective time intervals of the first and seconds stages, where $t_1 = t_0 + T, t_2 = t_1 + \tau$.

A. Definition of Safe State

A *safe state* is defined to be a state from which the second stage lane-change maneuver is guaranteed to be collision-free. Using the piecewise constant angular acceleration scheme (5), the lateral movement is symmetric as evident in Figure 4. Assume $L_L = 2L_y$, for otherwise the vehicle could keep on traveling in the middle of two lanes. In the first half of the maneuver duration, the vehicle moves in the source lane towards the middle point of the source and target lanes where θ reaches the maximum. In the second half the vehicle moves in the target lane while driving θ back to 0. The following three conditions in the second stage have to be met to be a safe state,

1) In $[t_1, t_1 + \frac{\tau}{2})$, the ego vehicle should not hit the leading vehicle.
2) In $[t_1 + \frac{\tau}{2}, t_1 + \tau)$, the ego vehicle should not hit any adjacent vehicle.
3) At $t_1 + \tau$, the ego vehicle should keep a safe distance from any adjacent vehicle.

Here an *adjacent vehicle* is any vehicle in the target lane. The union of the leading and adjacent vehicles are referred to as the *surrounding vehicles*. These three conditions are more precisely specified next.

B. Conditions of Safe State

Recall from Section II-C that the ego vehicle maintains a constant speed during the lane-change maneuver. In the worst case scenario of potential collision, the leading vehicle makes the maximum deceleration. The distance from the ego vehicle to the leading vehicle is given by

$$d_{e,l}(t) = d_{e,l}(t_1) + \big(s_l(t_1) - s_e(t_1)\big)\,t - \frac{1}{2}a_{\max}t^2. \quad (12)$$

Therefore the first condition states that $\min_{0\le t<\frac{\tau}{2}} d_{e,l}(t) \ge L_x$. Because the above distance function is concave, the minimum occurs at two boundary points. Since $d_{e,l}(0) \ge L_x$, it follows that

$$d_{e,l}\left(\frac{\tau}{2}\right) \ge L_x. \quad (13)$$

Next, the distance from the ego vehicle to an adjacent vehicle i is bounded by

$$d_{e,i,-}(t) \le d_{e,i}(t) \le d_{e,i,+}(t) \quad (14)$$

$$d_{e,i,-}(t) = d_{e,i}(t_1) + \big(s_i(t_1) - s_e(t_1)\big)\,t - \frac{1}{2}a_{\max}t^2 \quad (15)$$

$$d_{e,i,+}(t) = d_{e,i}(t_1) + \big(s_i(t_1) - s_e(t_1)\big)\,t + \frac{1}{2}a_{\max}t^2 \quad (16)$$

As before, a positive distance means vehicle i is ahead of the ego vehicle. The second condition states that in the entire interval $[t_1 + \frac{\tau}{2}, t_1 + \tau)$ vehicle i is either ahead of the ego vehicle,

$$d_{e,i,-}\left(\frac{\tau}{2}\right) \ge L_x \text{ and } d_{e,i,-}(\tau) \ge L_x, \quad (17)$$

or behind the ego vehicle,

$$-d_{e,i,+}\left(\frac{\tau}{2}\right) \ge L_x \text{ and } -d_{e,i,+}(\tau) \ge L_x \quad (18)$$

Finally, at t_2, $s_e(t_2) = s_e(t_1)$. In the worst case of potential collision, $s_i(t_2) = s_i(t_1) \pm a_{\max}\tau$, where the $\pm$ sign depends on whether vehicle i is ahead of or behind the ego vehicle. From (4), the third condition states that if vehicle i is ahead, then

$$d_{e,i,-}(\tau) \ge \max\left(0, \frac{s_e^2(t_1) - \big(s_i(t_1) - a_{\max}\tau\big)^2}{2a_{\max}}\right) + L_x, \quad (19)$$

and vehicle i is behind, then

$$-d_{e,i,+}(\tau) \ge \max\left(0, \frac{-s_e^2(t_1) + \big(s_i(t_1) + a_{\max}\tau\big)^2}{2a_{\max}}\right) + L_x, \quad (20)$$

In summary, (12) to (20) specify the conditions of being a safe state at t_1. These conditions are derived to ensure collision-free in the second stage lane-change maneuver in $[t_1, t_1+\tau)$. The next question, which is the focus of the following three sections, is how to optimally reach a safe state from the present state at t_0, i.e., to minimize T, the duration of the first stage longitudinal travel.

IV. Model-based Optimization

A. *Discrete-time Optimization Using Model-based Prediction*

Section III-B shows that whether a safe state is reached depends on the relative distance of the ego vehicle and the surrounding ones and their speeds at t_1, which depend on their kinetic behaviors in $[t_0, t_1)$. For simplicity, assume that in this interval all the surrounding vehicles travel longitudinally at constant speeds and that only the ego vehicle adjusts its acceleration $a_e(t)$. Section IV-C will address the scenario where the surrounding vehicles also adjust their accelerations.

In Section II-B, longitudinal travel is modeled as a continuous-time operation where control variable $a(t)$ is a function of continuous time t. To simplify the optimization, consider a discrete-time approximation where $a(t)$ can only change at discrete time instants $t_0, t_0+\delta, t_0+2\delta, \ldots$, where δ is the step size parameter. Suppose that $T = K\delta$, where $K \geq 0$ is integer. The objective of the optimization is to minimize K by finding the optimal acceleration sequence $a_e(0), a_e(1), \ldots, a_e(K-1)$, which are the accelerations of the ego vehicle at $t_0, t_0+\delta, \ldots, t_0+(K-1)\delta$, such that (12) to (20) are satisfied at $t_1 = t_0 + K\delta$ for the leading vehicle and all the adjacent vehicles.

To check these conditions, one has to calculate the distance between the ego vehicle and any surrounding one i, and their speeds at t_1, which are given as follows.

$$\begin{aligned} s_i(t_0+K\delta) &= s_i(t_0), && (21)\\ s_e(t_0+K\delta) &= s_e(t_0)+u(K)\delta, && (22)\end{aligned}$$

And $d_{e,i}(t_0+K\delta)$ is given by

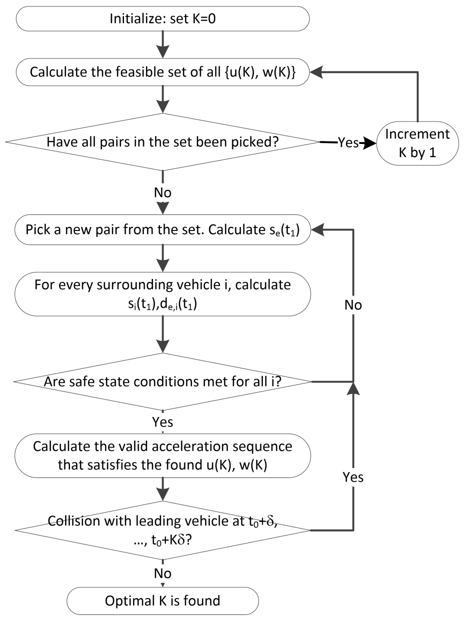
$$\begin{aligned} & d_{e,i}(t_0) + \left(\sum_{j=0}^{K-1} s_i(t_0+j\delta) - s_e(t_0+j\delta)\right)\delta && (23)\\ &= d_{e,i}(t_0) + \big(s_i(t_0)-s_e(t_0)\big)K\delta - w(K)\delta^2, && (24)\end{aligned}$$

where

$$\begin{aligned} u(K) &= \sum_{j=0}^{K-1} a_e(j), && (25)\\ w(K) &= \sum_{j=0}^{K-1}\left(\sum_{j'=0}^{j-1} a_e(j')\right). && (26)\end{aligned}$$

Clearly, only $u(K), w(K)$ in (21) to (24) depend on the acceleration sequence $a_e(k)$, $k = 0, \ldots, K-1$. Figure 7 shows the flow chart of the model-based optimization.

The main complexity of the optimization lies in the construction of the feasible set of all $\{u(K), w(K)\}$ pairs. Calculating $s_e(t_1), d_{e,i}(t_1)$ and checking the safe state conditions



Initialize: set K=0
Calculate the feasible set of all {u(K), w(K)}
Have all pairs in the set been picked?
Yes
Increment K by 1
No
Pick a new pair from the set. Calculate s_e(t_1)
For every surrounding vehicle i, calculate s_i(t_1),d_e,i(t_1)
No
Are safe state conditions met for all i?
Yes
Calculate the valid acceleration sequence that satisfies the found u(K), w(K)
Yes
Collision with leading vehicle at t_0+δ, ..., t_0+Kδ?
No
Optimal K is found

Fig. 7: Flow chart of model-based optimization.

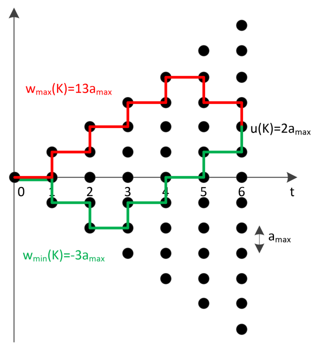


Fig. 8: Graphic illustration of calculating $u(K), w(K)$ along two different paths.

are straightforward from (12) to (24). However, because $a_e(k)$ can be any real number in $[-a_{\max}, a_{\max}]$, the feasible set has an infinite number of elements. Even if $a_e(k)$ is limited to a finite number of discrete numbers, with a brute-force approach, the size of the feasible set grows *exponentially* with K. To address this complexity challenge, the paper next exploits the structure in (25) and (26) and proposes an efficient algorithm to construct the feasible set where the search complexity grows *quadratically* with K.

B. *An Efficient Search Algorithm for Optimizing Lane-change Path Planning*

For simplicity, first limit $a_e(k)$ to only three possible values $0, \pm a_{\max}$. The general case of continuous $a_e(k)$ will be addressed later in this section.

Figure 8 depicts two possible paths of $a_e(k)$ as k increases from 0 to K. Because $u(k) = u(k-1) + a_e(k-1)$, $u(k-1)$ either steps up or down by $a_{\max}$ to arrive at $u(k)$ or remains unchanged depending on $a_e(k-1)$. Here $u(0) = 0$. $w(k)$ is the net (positive or negative) area between the corresponding path of $u(0), \ldots, u(K-1)$ and the x-axis.

The possible values of $u(K)$ is $\{-K, -(K-1), \ldots, 0, 1, \ldots, K\}a_{\max}$. For a given $u(K)$, the value of $w(K)$ is not unique. As an example, in Figure 8, $K = 6$. The two paths both lead to $u(6) = 2a_{\max}$. $w(6) = 13a_{\max}$ for the red path and $w(6) = -3a_{\max}$ for the green path. The following propositions characterize the feasible set of $w(K)$ for a given $u(K)$.

Proposition 4.1: For a given $u(K)$, the achievable $w(K)$ is bounded by $w_{\min}(K) \leq w(K) \leq w_{\max}(K)$. $w_{\min}(K)$ is achieved with

$$\begin{aligned} &a_e(0), \ldots, a_e(k_1 - 1) = -a_{\max}, \\ &a_e(k_1), \ldots, a_e(k_1 + k_2 - 1) = 0, \\ &a_e(k_1 + k_2), \ldots, a_e(K_1) = a_{\max}, \end{aligned} \tag{27}$$

where

$$k_1 = \left\lfloor \frac{K - \frac{u(K)}{a_{\max}}}{2} \right\rfloor, k_2 = K - \frac{u(K)}{a_{\max}} - 2k_1. \tag{28}$$

$w_{\max}(K)$ is achieved with

$$\begin{aligned} &a_e(0), \ldots, a_e(k_1' - 1) = a_{\max}, \\ &a_e(k_1'), \ldots, a_e(k_1' + k_2' - 1) = 0, \\ &a_e(k_1' + k_2'), \ldots, a_e(K_1) = -a_{\max}, \end{aligned} \tag{29}$$

where

$$k_1' = \left\lfloor \frac{K + \frac{u(K)}{a_{\max}}}{2} \right\rfloor, k_2' = K + \frac{u(K)}{a_{\max}} - 2k_1'. \tag{30}$$

Proof Rewrite (26) as

$$w(K) = \sum_{j=0}^{K-1} (K - 1 - j)a_e(j). \tag{31}$$

For a given $u(K)$, $\sum_{j=0}^{K-1} a_e(j)$ is fixed. In (31), the coefficients of $a_e(0), \ldots, a_e(K-1)$ are strictly decreasing. Thus, given the fixed total, the values of $a_e(0), \ldots, a_e(K-1)$ should be made in descending order to maximize $w(K)$ and in ascending order to minimize $w(K)$.

For $w_{\min}(K)$, the acceleration sequence $\{a_e(0), \ldots, a_e(K-1)\}$ should start with some $-a_{\max}$'s followed by some 0's and finally some $a_{\max}$'s, as in (27). Moreover,

$$\sum_{j=0}^{K-1} a_e(j) = u(K) \Rightarrow (K - k_1 - k_2) - k_1 = \frac{u(K)}{a_{\max}}. \tag{32}$$

Minimizing $w(K)$ is equivalent to maximizing k_1. Hence, (28) follows.

Similarly, for $w_{\max}(K)$, the acceleration sequence $\{a_e(0), \ldots, a_e(K-1)\}$ starts with some $a_{\max}$'s followed by some 0's and finally some $-a_{\max}$'s, as in (29).

$$\sum_{j=0}^{K-1} a_e(j) = u(K) \Rightarrow k_1' - (K - k_1' - k_2') = \frac{u(K)}{a_{\max}}. \tag{33}$$

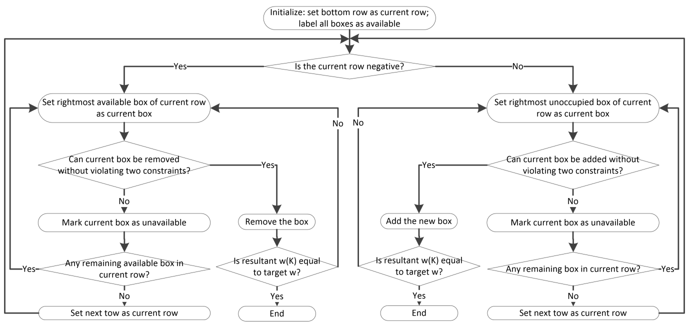

Initialize: set bottom row as current row; label all boxes as available
Is the current row negative?
Yes
No
Set rightmost available box of current row as current box
Can current box be removed without violating two constraints?
Yes
No
Mark current box as unavailable
Remove the box
Any remaining available box in current row?
Yes
No
Is resultant w(K) equal to target w?
Yes
No
Set next tow as current row
End
Set rightmost unoccupied box of current row as current box
Can current box be added without violating two constraints?
Yes
No
Add the new box
Mark current box as unavailable
Is resultant w(K) equal to target w?
Yes
No
Any remaining box in current row?
Yes
No
End
Set next tow as current row

Fig. 9: Flow chart of adjusting the path of $u(0), \ldots, u(K-1)$ to achieve target $w(K) = w$.

Maximizing $w(K)$ is equivalent to maximizing k_1'. Hence, (30) follows. ■

Proposition 4.2: For any w in discrete set $\{w_{\min}(K), w_{\min}(K) + 1, \ldots, w_{\max}(K)\}$, one can always find an acceleration sequence $\{a_e(0), \ldots, a_e(K-1)\}$ such that the resultant $w(K) = w$.

Proof Prove the proposition by construction. In Figure 8 the net area between a path of $u(0), \ldots, u(K-1)$ and the x-axis is the sum of small positive or negative rectangular boxes each of area $\pm a_{\max}$. Starting from the path corresponding to $w_{\max}(K)$, which is defined by the acceleration sequence given in Proposition 4.1, sequentially reduce the number of negative boxes and increase the number of positive boxes by adjusting the path so that the sum reaches the target w.

A valid path has to meet the following two constraints.

1) The two ending points are fixed where $u(0) = 0$ and $u(K)$ is given.
2) In every step, the path can go up or down by one step size $a_{\max}$, or remain unchanged.

Figure 9 describes a procedure of adjusting the path. Note that the procedure and thus the path that achieves $w(K) = w$ may not be unique. Figure 10 depicts an example for the scenario shown in Figure 8. ■

Propositions 4.1 and 4.2 state that the feasible set in Figure 7 is characterized as follows. $u(K)$ takes every element in $\{-K, -(K-1), \ldots, 0, 1, \ldots, K\}a_{\max}$. For a given $u(K)$, $w(K)$ takes every element in $\{w_{\min}(K), w_{\min}(K) + 1, \ldots, w_{\max}(K)\}$. Because $w_{\min}(K), w_{\max}(K)$ are in the order of K, the complexity of examining all feasible pairs of $\{u(K), w(K)\}$ grows with K^2.

One caveat is that if there is a leading vehicle, then the optimal $u(K)$, $w(K)$ have to ensure that it is not hit by the ego vehicle in $[t_0, t_1)$. Recall that the safe state conditions only impose the constraints at t_1. Therefore, the flow chart of Figure 7 requires to examine whether the found acceleration sequence that satisfies $u(K), w(K)$ do not cause any collision at $t_0+\delta, \ldots, t_0+K\delta$ assuming the leading vehicle drives at its present speed. As noted in the proof of Proposition 4.2 multiple acceleration sequences may achieve the same $u(K), w(K)$. In this case, examine the one that delays the acceleration

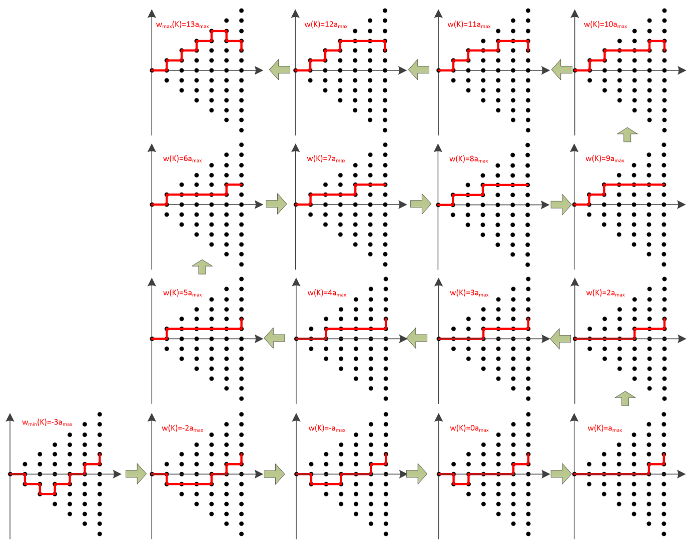


Fig. 10: An example of adjusting the path of $u(0), \ldots, u(K-1)$ to achieve various targets w given $u(K) = 2a_{\max}$, starting from the path corresponding to $w_{\min}(K)$ to the one corresponding to $w_{\max}(K)$.

as much as possible, because it minimizes the possibility of hitting the leading vehicle. This consideration motivates the search procedure described in the flow chart of Figure 9.

In the calculation of $w_{\min}(K), w_{\max}(K)$ so far, the constraints of maximum and minimum speeds $s_{\min}, s_{\max}$ have not been taken into account. To consider these constraints, the feasible u(K) is limited to

$$\frac{s_{\min} - s_e(t_0)}{\delta} \leq u(K) \leq \frac{s_{\max} - s_e(t_0)}{\delta}, \tag{34}$$

and k_1, k_1' obtained in (28) and (30) are limited to

$$k_1 \leq \frac{-s_{\min} + s_e(t_0)}{a_{\max}\delta}, k_1' \leq \frac{s_{\max} - s_e(t_0)}{a_{\max}\delta} \tag{35}$$

So far $a_e(k)$ has been limited to only three discrete values $0, \pm a_{\max}$. Now let $a_e(k)$ to take continuous value. Thus, $u(K), w(K)$ are also continuous. Propositions 4.1 requires modification. Take $w_{\max}(K)$ for example. Suppose that the target $u(K)$ is such that $u_-(K) < u(K) < u_-(K) + a_{\max}$, where $\frac{u_-(K)}{a_{\max}}$ is integer. By definition, $0 < u(K) - u_-(K) < a_{\max}$. First obtain the optimal path with respect to $u_-(K)$ according to Propositions 4.1, and then bump up a fraction of the path by an amount equal to $u(K) - u_-(K)$ to increase $w_{\max}$.

Figure 11 shows two examples. The red solid line represents the optimal path obtained with respect to $u_-(K)$ according to Propositions 4.1. The black dash line represents the modified one from the red path by exploiting the difference $u(K) - u_-(K)$. From (30), $k_2' = 0, 1$. Figure 11a is an example of $k_2' = 0$ where the acceleration sequence for $u_-(K) = 2a_{\max}$ is $\{a_{\max}, a_{\max}, a_{\max}, a_{\max}, -a_{\max}, -a_{\max}\}$. The modified acceleration sequence to achieve $w_{\max}(K)$ is

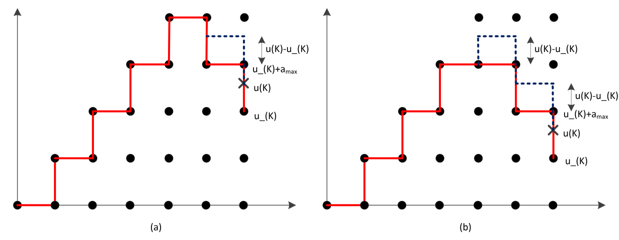


Fig. 11: An example of using continuous $a_e(k)$. Adjust the path of $u(0), \ldots, u(K-1)$ for a target $u(K)$ where $u_-(K) < u(K) < u_-(K) + a_{\max}$ and $\frac{u_-(K)}{a_{\max}}$ is integer.

$\{a_{\max}, a_{\max}, a_{\max}, a_{\max}, -a_{\max} + u(K) - u_-(K), -a_{\max}\}$. Figure 11b is an example of $k_2' = 1$ where the acceleration sequence for $u_-(K) = a_{\max}$ is $\{a_{\max}, a_{\max}, a_{\max}, 0, -a_{\max}, -a_{\max}\}$. The modified acceleration sequence to achieve $w_{\max}(K)$ is $\{a_{\max}, a_{\max}, a_{\max}, u(K) - u_-(K), -a_{\max}, -a_{\max}\}$.

The calculation of $w_{\min}$ can be modified similarly from Propositions 4.1. Propositions 4.2 also requires modification: any real number target $w \in [w_{\min}(k), w_{\max}(k)]$ can be achieved by an acceleration sequence.

C. Potential Problems

A major difficulty in path planning arises from the uncertainty of the future behaviors of the surrounding vehicles. Recall that Section III assumes the worst case scenario to calculate the conditions of a safe state. The rationale is that in the proposed two-stage path planning system, once in the second stage lane-change maneuver, the ego vehicle is migrating from the source to the target lane. The goal is to ensure *absolute safety* in this process so that the ego vehicle needs not to change its mind before it has safely arrived at the target lane.

On the other hand, this section assumes the surrounding vehicles travel at a constant speed in the planning of the first stage longitudinal travel. Assuming the worst case scenario here would be too conservative. Because in this stage the ego vehicle is still traveling in the source lane, it can adapt its planning by comparing the observed and assumed behaviors of the surrounding vehicles. Specifically, the optimization described in this section is not a one-shot process but a moving-horizon one. In the present time t_0, the ego vehicle carries out the two-stage path planning and obtains the optimal acceleration sequence $\{a_e(0), a_e(1), \ldots\}$. However, only $a_e(0)$ is applied at t_0. As time progresses to $t_0 + \delta$, the entire optimization is carried out again with the updated information of $s_i, d_{e,i}$ of all the surrounding vehicles i and the time horizon of the optimization advances by δ. Any error in the assumption of the surrounding vehicles is mitigated, as a new $a_e(0)$ is obtained and applied at time $t_0 + \delta$ instead of $a_e(1)$ that is calculated in the previous optimization done at t_0 and meant to represent the planned acceleration at $t_0 + \delta$.

While moving-horizon optimization ensures that the ego vehicle does not use obsolete information and results in any

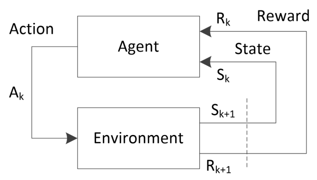


Fig. 12: Agent-environment interaction in RL.

safety issues, the path planning results may be severely suboptimal. For example, consider an adjacent vehicle i in the target lane next to the ego one, i.e., $d_{e,i} = 0$. Suppose that the optimization result is that the ego vehicle is to accelerate to surpass i and move to the target lane ahead of i. If vehicle i happens to decelerate, then the ego vehicle would find it easier than expected to surpass i. However, if vehicle i is also accelerating, then the ego vehicle may not be able to lead i by a safe distance for a long time. The optimal strategy with genie information about the behavior of vehicle i would be to decelerate and move to the target lane behind i.

This example clearly shows that the performance of the model-based optimization relies heavily on the accuracy of the model assumptions. The next section employs a model-free approach using deep reinforcement learning to overcome this problem.

V. Deep Reinforcement Learning

A. *Brief Introduction of Deep Reinforcement Learning*

Reinforcement learning (RL) is a machine learning approach for sequential decision making problems. Figure 12 depicts the basic diagram of RL. Denote S_k the state of the environment at discrete time instant k. Denote Φ the state space where $S_k \in \Phi$. For simplicity, assume that S_k can be observed by a RL agent. The RL agent takes an action A_k in the action space Ψ according to a policy π,

$$A_k = \pi(S_k). \tag{36}$$

Action A_k drives the environment to a new state S_{k+1} in the next time instant. The environment can be modeled by a state transition probability $P_{S_k,S}(A_k)$, the probability that $S_{k+1} = S \in \Phi$ given the present state S_k and action A_k. $P_{S_k,S}(A_k)$ represents the system model. As will be clear later, in a model-free algorithm the agent does not know $P_{S_k,S}(A_k)$.

The agent obtains a reward R_k as a function of $\{A_k, S_k\}$. The total discounted reward over time under policy π is

$$V^{\pi}(S_k) = R_k + \gamma R_{k+1} + \gamma^2 R_{k+2} + \cdots, \tag{37}$$

where $\gamma \in (0,1)$ is a discount factor.

The goal of RL is to find the optimal policy π^* that maximizes $V^{\pi}(S_k)$. This is more challenging than supervised learning used commonly in deep learning, where a set of training samples $\{S_k \to A_k\}$ have been given and the job of supervised learning is to use a neural network to memorize the training set and generalize to test data. In RL, such directed training samples are not available - for a given S_k, the "correct" answer A_k is not provided. The agent has to gradually figure out the optimal policy by interacting with the environment. In a sense, RL is to learn from experience rather than from a well defined training dataset.

A number of RL algorithms have been studied in the literature [8]. The paper next focuses on the model-free Q-Learning algorithm [9], [10]. The Q-Learning algorithm employs a so-called Q-function

$$Q^{\pi}(S_k, A_k) = R_k(S_k, A_k) + \gamma \sum_{S \in \Phi} P_{S_k,S}(A_k) V^{\pi}(S). \tag{38}$$

$Q^{\pi}(S_k, A_k)$ represents the total reward obtained by selecting A_k at time instant k and then following policy π in subsequent time instants $k+1, k+2, \ldots$. If function $Q^{\pi}(S, A)$ is known, then the optimal policy is obtained by optimizing $Q^{\pi}(S_k, A)$ over the action space Ψ,

$$A_k^* = \pi^*(S_k) = \arg\max_{A \in \Psi} Q^{\pi}(S_k, A). \tag{39}$$

For the optimal policy π^*, the actions at k and $k+1, k+2, \ldots$ are by definition all optimal. Therefore, for any $S \in \Phi$, the total discounted reward under π^* is obtained by optimizing $Q^{\pi}(S, A)$ over the action space Ψ,

$$V^{\pi^*}(S) = \max_{A \in \Psi} Q^{\pi^*}(S, A). \tag{40}$$

Therefore the recursive Bellman's optimality equation follows from (38) and (40)

$$\begin{aligned} Q^{\pi^*}(S_k, A_k) &= R_k(S_k, A_k) \\ &+\gamma \sum_{S \in \Phi} \left(P_{S_k,S}(A_k) \left(\max_{A \in \Psi} Q^{\pi^*}(S, A) \right) \right). \end{aligned} \tag{41}$$

(41) still relies on $P_{S_k,S}(A_k)$. To avoid the use of the model, the Q-Learning algorithm makes the following approximation,

$$\sum_{S \in \Phi} \left(P_{S_k,S}(A_k) \left(\max_{A \in \Psi} Q^{\pi^*}(S, A) \right) \right) \approx \max_{A \in \Psi} Q^{\pi^*}(S_{k+1}, A), \tag{42}$$

where S_{k+1} is an instance of the state resulting from action A_k. Recall from the system model that given S_k, A_k, the state S at $k+1$ is a random variable whose distribution is given by $P_{S_k,S}(A_k)$. (42) in effect ignores the probability distribution and focuses on one instance. When the learning is iterated over many time instants, many randomly generated instances will mimic the distribution due to the averaging effect therefore justifying the above approximation (42).

Convert (41) into a learning rule to update $Q^{\pi^*}(S_k, A_k)$ with (42),

$$Q^{\pi^*}(S_k, A_k) \Leftarrow Q^{\pi^*}(S_k, A_k) + \beta \Delta_{Q^{\pi^*}(S_k, A_k)}, \tag{43}$$

$$\begin{aligned} \Delta_{Q^{\pi^*}(S_k, A_k)} &= R_k(S_k, A_k) \\ &+\gamma \max_{A \in \Psi} Q^{\pi^*}(S_{k+1}, A) - Q^{\pi^*}(S_k, A_k), \end{aligned} \tag{44}$$

where $\Leftarrow$ in (43) means that the left side is updated by the right side. (43) and (44) are the key steps of the model-free Q-Learning algorithm.

One idea of deep RL called Deep Q-Learning (DQL) [11], [12] is to represent $Q^{\pi^*}(S, A)$ with a deep neural network

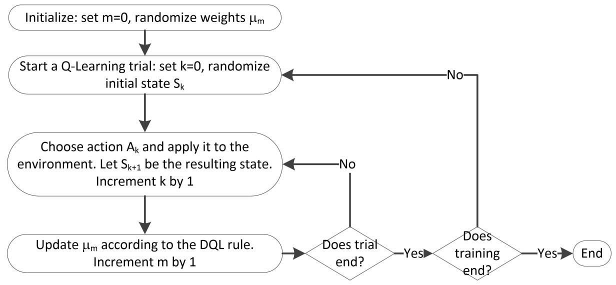

Initialize: set m=0, randomize weights μm
Start a Q-Learning trial: set k=0, randomize initial state Sk
Choose action Ak and apply it to the environment. Let Sk+1 be the resulting state. Increment k by 1
Update μm according to the DQL rule. Increment m by 1
Does trial end?
Does training end?
Yes
No
End

Fig. 13: Flow chart of DQL training.

$Q^{\pi^*}(S, A, \boldsymbol{\mu})$, where vector $\boldsymbol{\mu}$ contains the weights of the neural network, and use the RL learning rule to train the neural network weights $\boldsymbol{\mu}$. At time k, the loss function is a function of current weights $\boldsymbol{\mu}_k$,

$$L_{DQL}(\boldsymbol{\mu}_k) = \left(\Delta_{Q^{\pi^*}(S_k, A_k, \boldsymbol{\mu}_k)}\right)^2, \tag{45}$$

$$\Delta_{Q^{\pi^*}(S_k, A_k, \boldsymbol{\mu}_k)} = R_k(S_k, A_k) + \gamma \max_{A \in \Psi} Q^{\pi^*}(S_{k+1}, A, \boldsymbol{\mu}_k) - Q^{\pi^*}(S_k, A_k, \boldsymbol{\mu}_k). \tag{46}$$

Following (43), the updating rule of the weights is

$$\boldsymbol{\mu}_{k+1} = \boldsymbol{\mu}_k + \beta \left(\Delta_{Q^{\pi^*}(S_k, A_k, \boldsymbol{\mu}_k)}\right) \cdot \left(\nabla_{\boldsymbol{\mu}} Q^{\pi^*}(S_k, A_k, \boldsymbol{\mu}_k)\right), \tag{47}$$

where β is the learning rate and $\nabla_{\boldsymbol{\mu}_k} Q^{\pi^*}(S_k, A_k, \boldsymbol{\mu}_k)$ is the gradient of the output of the neural network with respect to weights $\boldsymbol{\mu}$. Figure 13 describes the DQL training. Once the training has converged, (39) generates the DQL policy.

B. Deep Q-Learning Algorithm of Lane-change Path Planning

To apply the DQL algorithm to the lane-change path planning problem, first define the state of the environment S_k to be a vector of the ego vehicle's speed and the surrounding vehicles' speeds and distances at the moment,

$$S_k = \left\{s_e(k), \left(s_l(k), d_{e,l}(t)\right), \left(s_i(k), d_{e,i}(t)\right), i = 1, 2, \ldots\right\}. \tag{48}$$

Note that the state space Φ is continuous.

The action in the DQL algorithm is a scalar, $A_k = a_e(k)$. For simplicity, the action space Ψ is assumed to be discrete where $A_k = \{-a_{\max}, 0, a_{\max}\}$. The only constraint is that the ego vehicle has to maintain a safe distance (4) from the leading vehicle at all time, which means that A_k may not take $a_{\max}$ or even 0 in some states.

The DQL algorithm itself is unaware of the state transition probability, which characterizes how the environment evolves, which is needed to run the simulation. For simplicity, the paper makes the following assumption for the simulation: except for the ego one, all vehicles intend to stay in their current lanes and their longitudinal driving behavior is specified with a well-developed car-following model, Intelligent Driver Model (IDM) [13]. In the IDM, the acceleration of a vehicle $\dot{s}$ is the following continuous function of its own speed s, the distance

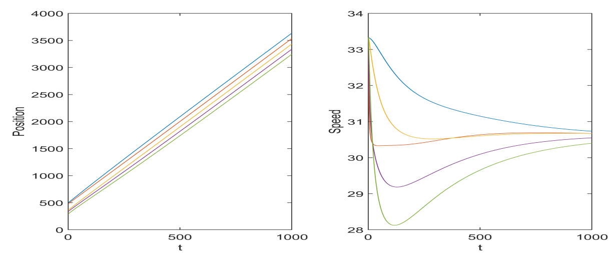


Fig. 14: An example of driving trajectories in IDM. The left figure shows the positions of the vehicles and the right figure shows the speeds.

gap from its leading vehicle d, and the speed difference (approaching rate) between the two vehicles Δs,

$$\dot{s} = \eta_1 \left(1 - \left(\frac{s}{\eta_2}\right)^{\eta_3} - \left(\frac{\eta_4 + \eta_5 s + \frac{s \Delta s}{\eta_6}}{d}\right)^2\right), \tag{49}$$

where $\eta_1, \ldots, \eta_6$ are model parameters and the recommended values are given in [13].

The IDM is not a stochastic mode but a deterministic one in the sense that except for $s_e(k)$, S_{k+1} is completely determined by S_k. Recall that $s_e(k)$ is a function of $A_{k-1} = a_e(k-1)$. In the DQL training, when A_{k-1} is applied to the environment, it affects $s_e(k)$, which then affects other elements of state S according to (49).

To get an sense of how the IDM works, consider a single-lane loop of length 500 meters. Five vehicles are placed in the loop with random initial positions and speeds. In simulation they all move in the clockwise direction and use the IDM to control their respective accelerations. Figure 14 shows the result of one simulation where their positions and speeds vary as the simulation progresses for 100 seconds. In this example, their initial speeds are very different with each other, two vehicles are in very close positions and the other three are close with one another. After a transient period, their speeds converge to some common number and their positions are spread out roughly uniformly in the loop.

In the DQL training, the ego vehicle may trigger the desire of lane-change at anytime during the simulation, in a transient or steady state period. Once triggered, the ego vehicle follows the DQL algorithm while all other ones continue following the IDM.

Unlike the model proposed in this study, the IDM does not have an explicit notion of a safe distance, because deceleration can be made arbitrarily large to avoid hitting the leading vehicle. The deceleration in the transient period in Figure 14 is quite large and may not be realistic in practice.

The goal of plan planning is to reach a safe state in as few steps as possible. Thus, the study proposes the following reward function,

$$R_k(S_k, A_k) = \begin{cases} 10, & \text{if a safe state is reached,} \\ -1, & \text{otherwise.} \end{cases} \tag{50}$$

The idea is that if a safe state has been reached, a positive reward is given to the current policy; otherwise, a step is considered wasted and the current policy is penalized by a negative reward. In Figure 13 a trial ends if a safe state is reached or a maximum number of simulation steps has passed.

Because $Q^{\pi^*}(S, A, \boldsymbol{\mu})$ is a general function whose characteristics are not known before, consider a fully connected (FC) deep neural network that has been shown great capability of general function approximation. An FC network represents a general purpose connection pattern and makes no assumptions about the features in the function to be approximated. The drawback is that it is very expensive in terms of memory and computation.

Every neuron in a layer is connected to every neuron in the previous layer, and each connection has its own weight. The output of a neuron o with M inputs is $o = f\left(\sum_{m=1}^{M} \mu_m i_m + \mu_0\right)$, where $i_1, \ldots, i_M$ are the M inputs, $\mu_0, \ldots, \mu_M$ are the bias and weights, and f is the activation function. The input layer consists of input neurons representing S_k, A_k. Observe that to improve the convergence of training, it is important to normalize the inputs of the input layer neurons to the interval of $(0, 1)$ before feeding them to the network. There is only one output layer representing the scalar value of Q-function. Multiple hidden layers are between the input and output layers. In this study, the activation function is $f(x) = x$ in the input and output layers and is the logistic sigmoid $f(x) = \frac{1}{1+\exp(-x)}$ in the hidden layers. $\nabla_{\boldsymbol{\mu}_k} Q^{\pi^*}(S_k, A_k, \boldsymbol{\mu}_k)$ in (47) can be obtained by the gradient back-propagation from the output to all the other layers.

An important step in the DQL training of Figure 13 is for the ego vehicle to choose an action A_k. There are two seemingly conflicting goals, namely exploration and exploitation. Exploitation is to probe a limited, but promising, region of the search space in the neighborhood of a promising solution that is known so far. Exploration is to probe a much larger portion of the search space to discover new promising solutions. To balance the trade-off between the two, choose an action according to the popular Boltzmann distribution. Denote $\psi_n, n = 1, 2, \ldots$ the elements in action space Ψ. Choose one of $\{\psi_n\}$ for A_k according to probability

$$p_n = \frac{\exp\left(Q^{\pi^*}(S_k, \psi_n, \boldsymbol{\mu}_k)/\Lambda\right)}{\sum_j \exp\left(Q^{\pi^*}(S_k, \psi_j, \boldsymbol{\mu}_k)/\Lambda\right)}, \tag{51}$$

where Λ is a positive real parameter (temperature) that controls the stochastic selection. A smaller Λ tends to prefer exploitation over exploration. In the DQL training, the value of Λ gradually drops to close to 0 as the simulation progresses.

Similarly, the learning rate β also decreases towards 0 as learning progresses to improve stability and help convergence.

C. Potential problems

One limitation of the above proposed DQL algorithm is that only a number of discrete actions are allowed, while in practice the ego vehicle can take a continuous number for its acceleration. There are deep RL algorithms that address continuous action spaces [12], which will not be explored here.

Convergence is a major practical concern. Although the convergence of the Q-Learning algorithm has been proven under certain conditions [10], the curse of dimensionality may result in excessive learning time and even not being able to converge to a stable policy in some cases. As pointed out before, RL and in particular deep RL are in general harder than supervised learning. Not surprisingly, compared with many neural network algorithms dealing with problems such as image classification, the DQL algorithm finds it more difficult to converge.

Generalization is yet another concern. The DQL learning uses the interaction with an environment where surrounding vehicles follow the IDM. The DQL algorithm itself does not depend on the IDM and can be used to learn other driving models. However, the risk always exists where some drivers behave differently from the trained models, thereby resulting in sub-optimality.

To address the above potential problems, the paper will study the use of 5G V2V to assist lane-change path planning.

VI. 5G V2V Cooperation

The schemes studied so far have assumed that the ego vehicle uses either model assumptions or prior training to guess the future behaviors of surrounding vehicles, and makes its lane-change path planning decision without any real-time exchange with them. Explicit communication with surrounding vehicles through 5G V2V to understand or influence their future behaviors can reduce uncertainty in the planning and therefore enhance the performance.

V2V communications is a form of radio-frequency (RF) communications similar to Wi-Fi or mobile cellular and allows vehicles to exchange information about their presence and driving intention. Two V2V technologies are competing in the market, namely Dedicated Short Range Communications (DSRC) based on Wi-Fi and Cellular-V2V (C-V2V) based on 4G LTE [14]. Both technologies broadcast basic safety messages of position and speed information. Recently, 5G V2V aims to drastically enhance C-V2V by supporting a variety of traffic types and adding new features such as enhanced mobile broadband, high reliability and low latency [15].

5G V2V can help lane-change path planning in the following two ways.

1) Trajectory sharing. Vehicle i informs the ego vehicle its planned future speeds $s_i(t_0 + j\delta), j = 0, \ldots, K-1$ via 5G V2V. As a result, the optimization scheme in Section IV does not have to assume all the surrounding vehicles travel at constant speeds. (23) still holds and (24) is revised to

$$d_{e,i}(t_0 + K\delta) = d_{e,i}(t_0) + \left(\sum_{j=0}^{K-1} s_i(t_0 + j\delta)\right)\delta$$
$$-s_e(t_0)K\delta - w(K)\delta^2. \tag{52}$$

The summation term is known because of 5G V2V. The optimization procedure of $u(K), w(K)$ remains the same. This is a form of passive cooperation, because vehicle i does not alter its planned future behavior.

2) Cooperative path planning. As a form of active cooperation, the ego and surrounding vehicles can jointly optimize their path planning. As an example, surrounding vehicles are all willing to adjust their speeds to further help the ego vehicle. (21) and (24) becomes

$$\begin{aligned} s_i(t_0+K\delta) &= s_i(t_0)+u_i(K)\delta, & (53)\\ d_{e,i}(t_0+K\delta) &= d_{e,i}(t_0)+\big(s_i(t_0)-s_e(t_0)\big)K\delta \\ &\quad +\big(w_i(K)-w(K)\big)\delta^2, & (54)\end{aligned}$$

where

$$\begin{aligned} u_i(K) &= \sum_{j=0}^{K-1} a_i(j), & (55)\\ w_i(K) &= \sum_{j=0}^{K-1}\left(\sum_{j'=0}^{j-1} a_i(j')\right). & (56)\end{aligned}$$

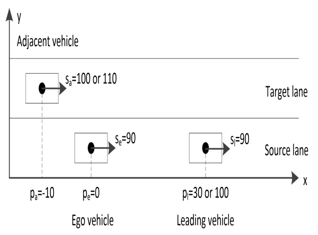
In addition to $u(K)$ and $w(K)$, $u_i(K), w_i(K)$ are variables for optimization. Clearly the complexity grows quickly as more surrounding vehicles participate in the joint optimization. In practice, only vehicles close to the ego vehicle will participate.

VII. Simulation Results and Discussion

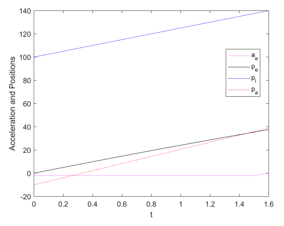
This section reports the numerical study of the three schemes presented in Sections IV to VI. The results are obtained via simulations.

The unit of all distances and positions is meter. The unit of time is second. The unit of all speeds is kilometer per hour. The unit of acceleration is meter squared per second. The following parameters are used in all the simulations: $L_x = 5, s_{\min} = 60, s_{\max} = 120, \tau = 1, a_{\max} = 2, \delta = 0.1$. The ego acceleration $a_e(k)$ takes three possible values $0, \pm a_{\max}$.

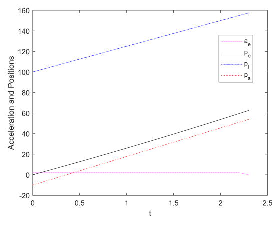
Consider the scenario of three vehicles: the ego vehicle, a leading one in the same lane and an adjacent one in the target lane. Their positions and speeds at $t_0 = 0$ are given in Figure 15a. The model-based optimization scheme is employed in the moving-horizon manner, as described in Section IV-C, to determine the ego vehicle's acceleration. The acceleration of the leading and adjacent vehicles is 0. Figure 15b, c and d plot the acceleration and the positions of the three vehicles as the simulation progresses for three cases of p_l, s_a. In Figure 15b, $p_l = 100, s_a = 110$. The optimal solution is to decelerate: $a_e(k) = -a_{\max}$ for $k = 0, 1, \ldots, 15$ so that at $t = 16\delta$ the ego vehicle is behind the adjacent vehicle and sufficiently slow to be at a safe state. In Figure 15c, the adjacent vehicle is slower: $p_l = 100, s_a = 100$. The optimal solution is completely opposite: to accelerate $a_e(k) = a_{\max}$ for $k = 0, 1, \ldots, 23$ to be sufficiently ahead of the adjacent vehicle and sufficiently fast at $t = 24\delta$. Finally in Figure 15d, the leading vehicle is closer: $p_l = 30, s_a = 100$. The presence of the nearby leading vehicle prevents the ego one from accelerating as far as in Figure 15c. The optimal solution is again completely opposite: to decelerate $a_e(k) = -a_{\max}$ for $k = 0, 1, \ldots, 24$ to be behind the adjacent vehicle and sufficiently slow at $t = 25\delta$.



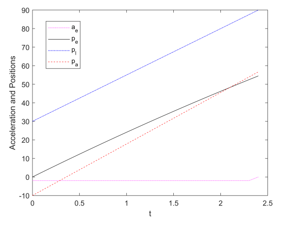
(a) Scenario illustration



(b) $p_l = 100, s_a = 110$

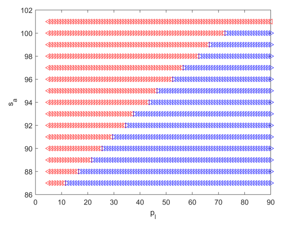


(a) $p_l = 100, s_a = 100$

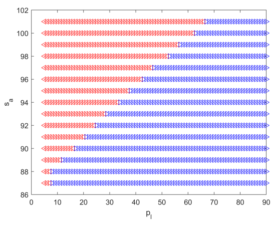


(b) $p_l = 30, s_a = 100$

Fig. 15: Trajectories of vehicle positions in three cases.



(a) $p_a = -10$



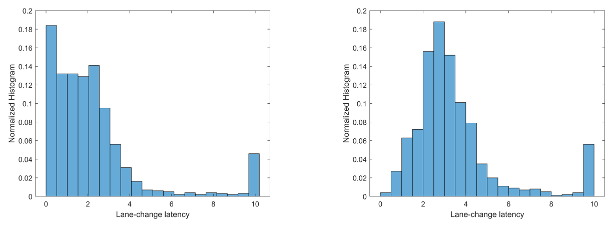
(b) $p_a = -20$

Fig. 16: Deceleration versus acceleration decision boundary.

Figure 15 seems to indicate[2] that the ego vehicle either accelerate or decelerate depending on p_l, s_a, p_a. Figure 16 depicts the acceleration versus deceleration decision map in two cases of $p_a = -10, -20$, where a red "<" indicates the decision is deceleration at given p_l, s_a and a blue ">" indicates acceleration. For given p_a, p_l, as s_a increases, the decision switches from acceleration to deceleration at some tipping point[3]. These tipping points form a decision boundary that clearly separates the areas of deceleration and acceleration. For a given p_a, the decision boundary can be approximately represented by a line of a positive slope between s_a and p_l. When $|p_a|$ increases from Figure 16a to Figure 16b,

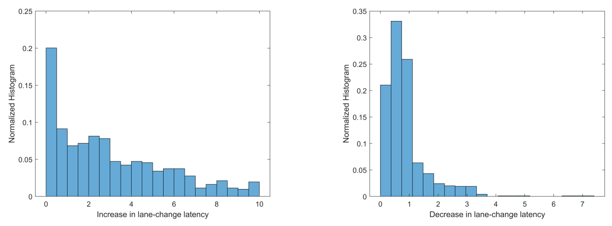
[2]It should be pointed out that the optimal sequence $a_e(k)$ is not always either $a_{\max}$ or $-a_{\max}$. For example, for $p_a = -10, p_l = 26, s_a = 90$, $a_e(k)$ is $\{-2, -2, -2, 0, \underbrace{2, \ldots, 2}_{11}\}$; for $p_a = -10, p_l = 30, s_a = 91$, $a_e(k) = \{\underbrace{2, \ldots, 2}_{7}, 0, 0, 0, 2, 0, 2\}$. For simplicity, both are labeled as acceleration in Figure 16 because the majority of the actions is acceleration.

[3]In Figure 16a for $p_a = -10, s_a = 101$, the decision is always deceleration irrespective of p_l.



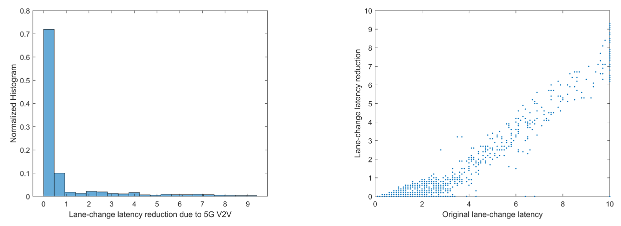
(a) 1 adjacent vehicle (b) 2 adjacent vehicles

Fig. 17: Normalized Histograms of lane-change latency with the model-based optimization. The height of a bar is the probability of the corresponding bin.



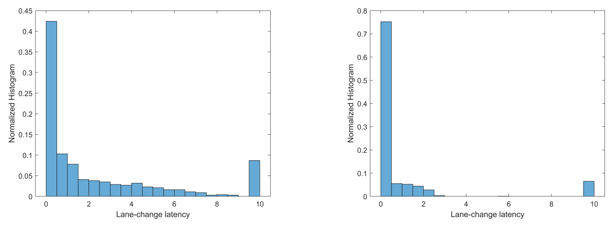
(a) Increase in latency (b) Decrease in latency

Fig. 18: Normalized Histograms of increase and decrease in lane-change latency due to variable speeds of adjacent vehicles.



(a) Normalized histogram (b) Scatter plot

Fig. 19: Statistics of reduction in lane-change latency due to 5G V2V.



(a) (b)

Fig. 20: Comparison of lane-change latency when adjacent vehicles follow the IDM. No leading vehicles are present. (a) Mode-based optimization and (b) Deep reinforcement learning.

the boundary line shifts to the left, meaning that for the same p_l, s_a, the decision tends to prefer acceleration over deceleration as the adjacent is farther behind.

The goal of path planning is to minimize the lane-change latency subject to the safety constraints. To study the statistical performance, uniformly randomly drop the surrounding vehicles, for 1000 times, in the following intervals: $p_l(t_0) \in (L_x, 20), p_a(t_0) \in (-10, 10), s_e(t_0) = s_l(t_0) \in (s_{\min}, s_{\max})$. $s_a(t_0)$ is randomly within $\pm 10\%$ of $s_e(t_0)$. Figure 17 plots the normalized histograms of the lane-change latency using the model-based optimization[4]. In Figure 17a, only one adjacent vehicle is dropped in the target lane. About half of the time, the ego vehicle can move to a safe state quickly within 2 second latency; however, the latency can be as large as 4 seconds or more in some cases. In Figure 17b two adjacent vehicles are dropped, which increases the congestion level and results in larger latency.

Section IV-C points out that the model-based optimization assumes surrounding vehicles travel at constant speeds and that its performance would be negatively or positively affected if this assumption does not hold. Figure 18 quantifies this effect statistically by plotting the normalized histograms of the increase and decrease in lane-change latency of variable speeds of adjacent vehicles as compared with that of constant speeds. The simulation is the same as in Figure 17a except that adjacent vehicles can accelerate or decelerate at $\pm a_{\max}$ subject to the $s_{\min}, s_{\max}$ constraints. Among the 1000 random samples, about 100 see no difference in lane-change latency and are thus excluded in the histogram of Figure 18. It is interesting to note that the distributions of the increase and decrease in latency are quite different. The increase in latency is more spread out in Figure 18a, while the distribution of the decrease is more concentrated in smaller values in Figure 18b, indicating that the negative impact due to variable speeds can be more profound than the positive one.

Figure 19 provides the statistics of the reduction in lane-change latency when 5G V2V is employed to better predict the future behaviors of adjacent vehicles. The results show that the improvement is insignificant in many sample points whose latency values are not large to start with. However, in other sample points with large latency values without 5G V2V, the improvement can be substantial. This means that the value of 5G V2V lies in helping the tail, i.e., the worst case, performance where help is most needed.

Finally, consider a different scenario where adjacent vehicles follow the IDM instead of simply accelerating ($a_{\max}$) or decelerating ($-a_{\max}$). Specifically, first simulate the IDM as in Figure 14 to get a sample set of $5 \cdot 10^4$ vehicle speed traces. For example, there are 5 speed traces in Figure 14b. Partition the sample set into a training set (80%) and a test set (20%). For simplicity, consider one adjacent vehicle and

[4]In the simulation, the maximum search window is set to $K = 100$. In some cases, the optimization algorithm cannot find any feasible sequence $a_e(k)$ to reach a safe state within $K \leq 100$. These instants are labeled as lane-change latency 10 seconds in Figure 17. This is the reason that the bar corresponding to 10 exhibits a spike.

no leading vehicle. The adjacent vehicle randomly picks one speed trace from the training set as its driving speed and $p_a(t_0) \in (-10, 10)$ as its initial position. The deep reinforcement learning DQN employs a network of two hidden layers, each of 30 nodes, and is trained with the training set. The DQN parameters are as follows. $\gamma = 0.8$. The number of training iterations is 10^6. The batch size of training is 100. The learning rate $\beta = \max(0.15 - 0.0001k, 0.05)$ and the temperature $\Lambda = 1.1^{-k}$ where $k = 1, 2, \ldots$ is simulation step index. The trained DQN is then used to determine a_e when the adjacent vehicle picks its driving speed from the test set. Figure 20b shows the normalized histogram of lane-change latency for the test set. As a comparison, Figure 20a shows the performance of the model-based optimization, which assumes (incorrectly) a constant speed of the adjacent vehicle in the prediction algorithm and does not enjoy the benefits of prior training. The DQN algorithm exhibits clear advantage in handling the scenario where the adjacent vehicle varies the speed.

VIII. Conclusion

This paper has studied lane-change path planning, a crucial and yet complex task in autonomous driving. The paper proposes that the entire path planning consists of two stages to address the requirements of both safety and efficiency. In the first stage the ego vehicle controls only the linear acceleration to reach a safe state. In the second stage the ego vehicle makes lateral maneuver by controlling the angular acceleration. The safe state conditions are derived based on a fixed pattern of lateral maneuver to ensure collision free in the second stage. Therefore, the lane-change path planning problem is converted to an optimization problem of minimizing the time to reach a safe state in the first stage. The paper proposes three schemes to determine the acceleration sequence, namely, kinetic model based optimization, deep reinforcement learning, and 5G vehicle-to-vehicle (V2V). The paper investigates these schemes via simulation. The model-based optimization is sensitive to the model assumptions. The deep reinforcement learning performs better than the model-based optimization when the actual driving behavior deviates from the model assumptions. The 5G V2V further enhances the robustness by eliminating uncertainty in predicting driving behavior behaviors by explicit communications among vehicles to share driving intents and enable cooperative driving.

Several areas can be considered for the future work.

- So far the safe state analysis only considers the leading vehicle in the source lane and the adjacent vehicles in the target lane. The vehicles in the lane left of the target lane should also be included in the analysis, because they may make lane-change maneuver to the target lane while the ego vehicle is making maneuver.
- While the deep reinforcement learning DQN algorithm and 5G V2V algorithm show advantages over the model-based optimization, the robustness should be thoroughly investigated. For the DQN algorithm, both the training and test sets come from the same IDM in my simulation. It would be interesting to see how the DQN algorithm performs when the test set is generated from a different driving model. For the 5G V2V algorithm, this paper assumes information sharing is perfect among surrounding vehicles. One should simulate the scenario where a fraction of V2V messages are lost.
- This paper focuses on one autonomous vehicle and assumes all other vehicles follow traditional driving models. It would be interesting to see how multiple autonomous vehicles interact among each other. The cooperative path planning briefly discussed in Section VI is one example. In practice a likely scenario is that autonomous vehicles will be mixed with non-autonomous ones and different autonomous vehicles may run different lane-change path planning algorithms. The analysis of such a mixed scenario is definitely more complex but will provide valuable insights in the real world.

References

[1] A. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, pp. 70–76, 01 2017.

[2] S. Hetrick, "Examination of driver lane change behavior and the potential effectiveness of warning onset rules for lane change or "side" crash avoidance systems," Ph.D. dissertation, Virginia Tech, 1997.

[3] K. I. Ahmed, "Modeling drivers' acceleration and lane changing behavior," Ph.D. dissertation, Massachusetts Institute of Technology, 1999.

[4] C. Vallon, Z. Ercan, A. Carvalho, and F. Borrelli, "A machine learning approach for personalized autonomous lane change initiation and control," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 06 2017, pp. 1590–1595.

[5] Y. Chen, "Learning-based lane following and changing behaviors for autonomous vehicle," Master's thesis, Carnegie Mellon University, Pittsburgh, PA, May 2018.

[6] C.-J. Hoel, K. Wolff, and L. Laine, "Automated speed and lane change decision making using deep reinforcement learning," 11 2018, pp. 2148–2155.

[7] P. Wang, C.-Y. CHAN, and A. De La Fortelle, "A Reinforcement Learning Based Approach for Automated Lane Change Maneuvers," in *IEEE Intelligent Vehicles Symposium*. Chang Shu, China: IEEE, 06 2018. [Online]. Available: https://hal-mines-paristech.archives-ouvertes.fr/hal-01980542

[8] S. Sathiya Keerthi and B. Ravindran, "A tutorial survey of reinforcement learning," *Sadhana*, vol. 19, no. 6, pp. 851–889, 12 1994. [Online]. Available: https://doi.org/10.1007/BF02743935

[9] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, Cambridge University, 01 1989.

[10] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 05 1992. [Online]. Available: https://doi.org/10.1007/BF00992698

[11] M. Riedmiller, "Neural fitted Q iteration – first experiences with a data efficient neural reinforcement learning method," in *Machine Learning: ECML 2005*, J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, and L. Torgo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 317–328.

[12] V. Franois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Foundations and Trends in Machine Learning*, vol. 11, no. 3-4, pp. 219–354, 2018. [Online]. Available: http://dx.doi.org/10.1561/2200000071

[13] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, pp. 1805–1824, 02 2000.

[14] T. Bey and G. Tewolde, "Evaluation of dsrc and lte for v2x," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, 01 2019, pp. 1032–1035.

[15] S. Chen, J. Hu, Y. Shi, Y. Peng, J. Fang, R. Zhao, and L. Zhao, "Vehicle-to-everything (v2x) services supported by lte-based systems and 5g," *IEEE Communications Standards Magazine*, vol. 1, no. 2, pp. 70–76, 2017.