Trajectory Planning for Automated Driving Based on Ordinal Optimization

Xiaoxin Fu, Yongheng Jiang*, Dexian Huang, Kaisheng Huang, and Jingchun Wang

Abstract: This paper proposes an approach based on Ordinal Optimization (OO) to solve trajectory planning for automated driving. As most planning approaches based on candidate curves optimize the trajectory curve and the velocity profile separately, this paper formulates the problem as an unified Non-Linear Programming (NLP) model, optimizing the trajectory curve and the acceleration profile (acceleration is the derivative of velocity) simultaneously. Then a hybrid optimization algorithm named OODE, developed by combining the idea of OO and Differential Evolution (DE), is proposed to solve the NLP model. With the acceleration profile optimized “roughly”, OODE computes and compares “rough” (biased but computationally-easier) curve evaluations to select the best curve from candidates, so that a good enough curve can be obtained very efficiently. Then the acceleration profile is optimized again “accurately” with the selected curve. Simulation results show that good enough solutions are ensured with a high probability and our method is capable of working in real time.

Key words: ordinal optimization; trajectory planning; automated driving; autonomous vehicle; rough evaluation

1 Introduction

In the past three decades, Advanced Driver Assistance Systems (ADAS) have attracted great interest from both academia and industry, backed by advances in sensing and computing technologies[1]. ADAS provide enormous benefits to transportation industry and our daily lives, such as increased safety, simplified driving tasks, and higher road utilization. Recently, more researches are devoted to developing fully autonomous vehicles[2–4], representing a higher level of automation. Trajectory planning, which generates a sequence of feasible movement states for the vehicle to maneuver amongst obstacles from an initial state to a desired terminal state, taking into account the vehicle’s kinematic model[5], plays an important and fundamental role in ADAS and autonomous vehicles.

There are two main challenges in trajectory planning. One is the computational complexity. As the vehicle moves in a dynamic environment, the path and the velocity profile both need to be planned. Since trajectory efficiency, comfort, safety, and economy should be optimized, time-consuming trajectory performance evaluation is also inevitably introduced. The other challenge is the high planning speed requirement. The vehicle is surrounded by multiple static or moving obstacles such as automobiles, bicycles, and pedestrians, whose future motion is uncertain. Trajectory planning should be performed in real time according to the fast-changing environment.

With above challenges, computational efficiency becomes the key to solving trajectory planning for automated driving.

1.1 Related work

We present some popular trajectory planning approaches as follows. Their efficiency (the quality of the returned solution and the real-time performance) will be commented on. Potential field methods[6–7]
could easily produce a collision-free path in a short time, but they are mainly designed for static or low-speed environment as potential fields are unsuitable for representing moving obstacles. These approaches also suffer from local optimum and lack the ability to accurately adjust the path smoothness.

Some methods based on optimal control[8, 9] use a predictive model to describe vehicle dynamics and do planning by solving a trajectory cost minimization problem with respect to control input. Because of the nonlinear kinematic constraints and the high demand for planning speed, the objective function for optimization is usually set to be quadratic forms of trajectory characteristic variables (e.g., steering angle and acceleration). But these methods are still not fast enough for planning in dynamic environment.

State lattice methods[10, 11] and sampling-based methods[5, 12] both build a graph of primitive trajectories, and then apply graph search to find the optimal trajectory. Although the dynamic environment can be modeled by adding the time dimension to the vehicle’s state space, the resultant search space grows exponentially. Due to the inherent inefficiency of searching, the applications of these methods are restricted.

Some other methods[13, 14] do the planning by optimizing the parameters in the function (e.g., polynomials and splines) that represents a trajectory. These approaches are mainly designed for applications with high real-time requirements, such as evasive maneuvers. But the solution is suboptimal since the trajectory is optimized on a particular type of functions. And accurate trajectory execution would be not easy since vehicle dynamics is not modeled.

1.2 Motivation

In real applications, trajectory planning is repeated in a receding horizon fashion. Each time only the trajectory segment in the first planning cycle is implemented. Since any vehicle maneuver can be seen as a combination of lane keeping and lane change behaviors, a limited number of candidate curves are thought to be enough for covering all possible paths for one single planning. Thus, in order to balance the model complexity and solution quality, some researchers developed candidate-curve-based planning approaches. A set of candidate trajectory curves that end with various states are first generated with the vehicle kinematic model. Then the best curve is selected from the candidates and combined with the velocity profile to form the trajectory solution.

Autonomous vehicle “Boss” assigns linear velocity profiles to candidate curves to produce candidate trajectories, from which the best trajectory is selected[2]. Robotic vehicle “Junior” selects the best curve based on the time cost plus a global cost, and then specifies the target driving speed[15]. In Ref. [16], the ranking of 6 maneuvers is output based on fast risk evaluation. Then in the allowed computation time, the planning result is produced by comparing trajectories within each maneuver according to a finer evaluation. These methods have been successfully applied on autonomous vehicles. However, since they either use a linear velocity profile or directly specify the target speed, a careful optimization of the velocity profile is lacked. Also because the trajectory curve and velocity are optimized separately, the obtained solution is suboptimal.

In this paper, we formulate the planning problem as an unified Non-Linear Programming (NLP) model, where the trajectory curve and the acceleration profile (acceleration is the derivative of velocity) are optimized simultaneously. Based on Ordinal Optimization (OO), we design a novel intelligent hybrid optimization algorithm named OO-based Differential Evolution (OODE) to solve the NLP model. Simulation results show the real time performance and solution quality of the proposed method.

The rest of this paper is organized as follows. Section 2 gives the modeling of the vehicle trajectory. Section 3 introduces the NLP model for trajectory planning. Section 4 talks about the basic idea of OO and proposes OODE. In Section 5, our method is simulated to plan trajectories among moving vehicles. The efficiency of OODE is compared with other two algorithms. Conclusion is made in the last section.

2 Vehicle Trajectory Model

The vehicle trajectory is modeled as two parts: the trajectory curve and the acceleration profile. The trajectory curve represents the movement of the vehicle reference point in the road plane. The acceleration profile characterizes how the vehicle acceleration changes along a given trajectory curve.

Based on the unicycle kinematic model, the vehicle pose \((x, y, \theta, \kappa)\) along the trajectory curve, which consists of position coordinates \((x, y)\), orientation \(\theta\), and curvature \(\kappa\), is described by the following four coupled, non-linear equations (parameterized by
distance \( s \) \(^{[17]}\):

\[
\begin{align*}
  x(s) &= \int \cos \theta(s) \, ds, \\
  y(s) &= \int \sin \theta(s) \, ds, \\
  \theta(s) &= \int \kappa(s) \, ds, \\
  \kappa(s) &= u(s)
\end{align*}
\]

where \( 0 \leq s \leq s_f \), \( s_f \) is the length of the curve.

Like Ref. \([17]\), a \( \tau \)-th-order polynomial is used to express curvature \( \kappa \) in terms of arc length \( s \):

\[
\kappa(s) = \kappa_0 + b_1 \cdot s + b_2 \cdot s^2 + \cdots + b_\tau \cdot s^\tau
\]

where \( \kappa_0 \) is the curvature at the curve’s start point \((s = 0)\), \( b_1, b_2, \ldots, b_\tau \) are polynomial coefficients. Let \( \mathbf{x}_0, \mathbf{y}_0, \theta_0, \kappa_0 \) denote the vehicle pose at the start point. The pose vector \( \mathbf{s} = [x \ y \ \theta \ \kappa]^T \) at any point in the curve can be computed as

\[
\begin{align*}
  \kappa(s) &= \kappa_0 + b_1 \cdot s + \cdots + b_\tau \cdot s^\tau, \\
  \theta(s) &= \theta_0 + \int_0^s \kappa(r) \, dr, \\
  x(s) &= x_0 + \int_0^s \cos \theta(r) \, dr, \\
  y(s) &= y_0 + \int_0^s \sin \theta(r) \, dr
\end{align*}
\]

The vehicle pose \([x_f \ y_f \ \theta_f \ \kappa_f]^T\) at the curve’s end point is obtained by plugging \( s = s_f \) into Eq. (3).

With given \([x_0 \ y_0 \ \theta_0 \ \kappa_0]^T\), \( \kappa(s) \) determines the geometry of the trajectory curve. Thus we combine the coefficients in \( \kappa(s) \) and curve length \( s_f \) to form a vector \( \mathbf{p} = [b_1 \ b_2 \ \cdots \ b_\tau \ s_f]^T \mathbf{p} \in \mathbb{R}^{\tau+1} \), which is defined to be the parameter vector of the trajectory curve.

The vehicle’s acceleration along a given trajectory curve is represented by a vector \( \mathbf{a} \). The curve is divided into \( N \) segments of equal length, and the vehicle is assumed to move at constant acceleration within each segment. Let \( \mathbf{a} = [a_1 \ a_2 \ \cdots \ a_N]^T \) store the accelerations \( a_1, a_2, \ldots, a_N \) in \( N \) segments. Then \( \mathbf{a} \ (a \in \mathbb{R}^N) \) determines the vehicle’s velocity at any point in the trajectory.

With the vehicle’s initial velocity \( v_0 \) and acceleration \( a_0 \), we can compute the vehicle’s velocity \( v_n \), curvature \( \kappa_n \), orientation angle \( \theta_n \), and position coordinates \( (x_n, y_n) \) at the end of the \( n \)-th trajectory segment by applying Simpson’s rule

\[ [f(a) + 4f((a + b)/2) + f(b)]/(b - a) \cdot f(x) \, dx \approx \int_a^b \]

where \( \Delta s_i = s_i/s_f \) is the length of each trajectory segment.

From Eqs. (3) and (4), \( \mathbf{p} \) and \( \mathbf{a} \) together determine the vehicle’s motion state \([x_n, y_n, \theta_n, \kappa_n, v_n, a_n] \) at \( N \) discrete points in the trajectory.

\section{Problem Formulation}

A vehicle trajectory can be denoted by a combination of the curve parameter vector \( \mathbf{p} \) and acceleration vector \( \mathbf{a} \). Therefore, this paper attempts to plan the trajectory by optimizing \( (\mathbf{p}, \mathbf{a}) \). To reduce the search space, we generate a set of candidate trajectory curves, based on which the planning problem is formulated as an NLP model.

\subsection{Generating candidate trajectory curves}

Because \( p \in \mathbb{R}^{\tau+1}, \mathbf{a} \in \mathbb{R}^N \), the vehicle trajectory’s solution space grows exponentially with \( \tau \) and \( N \) increasing. To narrow the search, we generate \( G \) candidate trajectory curves \( \Gamma_1, \Gamma_2, \ldots, \Gamma_G \), from which the best curve is selected. Thus the optimal value of \( \mathbf{p} \) is taken from set \([p_1, p_2, \ldots, p_G] \), where \( p_g \ (g = 1, 2, \ldots, G) \) is the parameter vector of curve \( \Gamma_g \). Each candidate curve \( \Gamma_g \) is defined to be the optimal curve that leads the vehicle to reach a predefined goal pose \( s_g \). By appropriately sampling \( s_g \) in the pose vector space, picking the best curve from \( \Gamma_1, \Gamma_2, \ldots, \Gamma_G \) could provide an acceptable good trajectory curve for execution.

The goal pose vectors are computed according to the driving environment and/or the driving decision from upper-layer planning modules. As an example, the goal poses used here are shown in Fig. 1. \( S \) is the trajectory start point. In \( s_g = [x_g^e \ y_g^e \ \theta_g^e \ \kappa_g^e]^T \), \( \theta_g^e = 0, \kappa_g^e = 0 \). \{\( x_g^e, y_g^e \) : \( g = 1, 2, \ldots, G \)\} are the positions of \( G \ (G = 2R + 1) \) goal points, which are composed of \( E_1, E_2, \ldots, E_R \) located in the left lane centerline, \( E_{R+1}, E_{R+2}, \ldots, E_{2R} \) located in the right lane centerline, and \( E_{2R+1} \) located in the middle lane.

![Fig. 1 An example of goal poses for trajectory planning.](image-url)
centerline. The goal points are distributed with equal spacing $\Delta l$. $l_{\text{plan}}$ is the longitudinal distance between the distribution center of the goal points and $S$. $d_{\text{plan}}$ is equal to the lane width. Then $(x^e_g, y^e_g)$ can be computed with $\Delta l$, $l_{\text{plan}}$, and $d_{\text{plan}}$.

The performance of the curve determined by $p$ is evaluated by a function $J_{\text{path}}(p)$, which is defined to be the objective function for curve optimization. $J_{\text{path}}(p)$ is the weighted sum of several costs (called as path costs), including length cost $C_{\text{Leng}}$, curvature cost $C_{\text{pCurv}}$, curvature derivative cost $C_{\text{dCurv}}$, and centerline offset cost $C_{\text{Offs}}$:

$$ J_{\text{path}}(p) = w_{\text{Leng}}C_{\text{Leng}} + w_{\text{pCurv}}C_{\text{pCurv}} + w_{\text{dCurv}}C_{\text{dCurv}} + w_{\text{Offs}}C_{\text{Offs}} $$

The computing formulas and impacts on trajectories of path costs are listed in Table 1, where $\kappa^n_\text{w}$ is the lane centerline curvature at the point closest to the end point of the $n$-th trajectory segment.

In Table 1, $C_{\text{Leng}}$ is the curve length, which affects driving efficiency. $C_{\text{pCurv}}$ is computed as the integral of squared curvature along the curve, since frequent sharp turns can affect both efficiency and comfort. $C_{\text{dCurv}}$ is computed as the integral of squared curvature derivative along the trajectory curve, because frequent changes in curve curvature can lower down comfort. $C_{\text{Offs}}$ is defined to consider the offset between the trajectory curve and the reference lane centerline, thus is computed as the sum of squared curvature differences.

Then the curve parameter vector $p_g$ is obtained by optimizing $J_{\text{path}}(p)$ in the following curve optimization model.$$
\Psi_g : \min J_{\text{path}}(p), \quad p \in \mathbb{R}^{t+1}$$

$$\text{s.t.} \quad g_i(s_g, p) = 0, \quad i = 1, 2, 3, 4,$$

$$l(p) > 0$$

where $l(p) = s_f, g_1(s_g, p) = x_f - x^e_g, g_2(s_g, p) = y_f - y^e_g, g_3(s_g, p) = \theta_f - \theta^e_g, g_4(s_g, p) = \kappa_f - \kappa^e_g$. $g_i(s_g, p) = 0$ ($i = 1, 2, 3, 4$) are introduced to guarantee that $\Gamma_g$ ends with pose $s_g$.

We introduce Lagrange multipliers and apply Newton’s method to solve $\Psi_g$. $s_f > 0$ is satisfied by setting an appropriate initial value of $p$. The solution of $\Psi_g$ could be obtained within 10 iterations. For more details about the initialization of $p$ and the derivation of iterative formulas, please refer to Ref. [17].

By solving $\Psi_g$, for $g = 1, 2, \ldots, G$, we get the parameter vectors of $G$ candidate trajectory curves, i.e., $p_1, p_2, \ldots, p_G$, which are stored in set $P$.

### 3.2 Trajectory optimization model

For a trajectory determined by $(p, a)$, since $J_{\text{path}}(p)$ describes the performance related to the trajectory curve (i.e., static performance), we further design $J_{\text{drive}}(p, a)$ to evaluate the dynamic performance, so that a comprehensive trajectory evaluation can be obtained by combining $J_{\text{path}}(p)$ and $J_{\text{drive}}(p, a)$. $J_{\text{drive}}(p, a)$ is the weighted sum of driving costs, including time cost $C_{\text{Time}}$, acceleration cost $C_{\text{pAcc}}$, acceleration increment cost $C_{\text{dAcc}}$, velocity cost $C_{\text{Spd}}$, and collision cost $C_{\text{Coll}}$:

$$ J_{\text{drive}}(p, a) = w_{\text{Time}}C_{\text{Time}} + w_{\text{pAcc}}C_{\text{pAcc}} + w_{\text{dAcc}}C_{\text{dAcc}} + w_{\text{Spd}}C_{\text{Spd}} + w_{\text{Coll}}C_{\text{Coll}} $$(7)

The computing formulas and impacts on trajectories of driving costs are given in Table 2. $C_{\text{Time}}$ is the time consumed for executing the whole trajectory, which affects efficiency. $C_{\text{pAcc}}$ is computed as the sum of squared accelerations in $N$ trajectory segments because both hard acceleration and deceleration reduce efficiency, safety, and economy. As the acceleration derivative indicates the jerk, $C_{\text{dAcc}}$ is computed as the sum of squared acceleration increments. $C_{\text{Spd}}$ and $C_{\text{Coll}}$ are defined to characterize the safety risk introduced by illegal velocity and potential collisions, respectively.

Obviously, $C_{\text{Time}}$, $C_{\text{pAcc}}$, and $C_{\text{dAcc}}$ can be calculated with $a$ and $s_f$. For $C_{\text{Spd}}$, $\eta_n$ is defined as

| Table 1 Path costs. |
|---------------------|
| Cost | Computing formula | Impact on trajectories |
| $C_{\text{Leng}}$ | $s_f$ | Efficiency |
| $C_{\text{pCurv}}$ | $\int_{0}^{s_f} \kappa(s)^2 ds$ | Efficiency, comfort |
| $C_{\text{dCurv}}$ | $\int_{0}^{s_f} \dot{\kappa}(s)^2 ds$ | Comfort |
| $C_{\text{Offs}}$ | $\sum_{n=1}^{N} (\kappa^n_{\text{w}} - \kappa^n_g)^2 \Delta s_f$ | Safety |

| Table 2 Driving costs. |
|-------------------------|
| Cost | Computing formula | Impact on trajectories |
| $C_{\text{Time}}$ | $I_N$ | Efficiency |
| $C_{\text{pAcc}}$ | $\sum_{n=1}^{N} a^n_{\text{a}} \Delta s_f$ | Comfort, safety, economy |
| $C_{\text{dAcc}}$ | $\sum_{n=1}^{N} (a^n_{\text{a}} - a^{n-1}_{\text{a}})^2 \Delta s_f$ | Comfort, economy |
| $C_{\text{Spd}}$ | $\sum_{n=1}^{N} \eta_n \Delta s_f$ | Safety |
| $C_{\text{Coll}}$ | $\sum_{n=1}^{N} v^n \Delta s_f$ | Safety |
\[
\eta_n = \begin{cases} 
1, & \text{if } v_n > v_{spd} \text{ or } v_n < 0; \\
0, & \text{otherwise} 
\end{cases}
\]  
(8)

where \(v_{spd}\) is the largest allowed velocity. In \(C_{Coll,n}\), \(e_n\) is calculated as \(e_n = e_{a,n} + e_{\beta,n} + \cdots\), where \(e_{a,n}, e_{\beta,n}, \ldots\) are the risks of the \(n\)-th trajectory segment due to potential collisions between the Ego Vehicle (EV) and the obstacles \(a, \beta, \ldots\), respectively. As an example, the calculation of \(e_{a,n}\) is depicted as follows.

The body of EV is approximated by \(K\) circles \(\Phi_1, \Phi_2, \ldots, \Phi_K\) of equal radius \(r_C\), which are located along EV’s body centerline with a fixed spacing \(\Delta r_C\) (see Fig. 2). With EV’s motion state \((x_n, y_n, \theta_n, \kappa_n)\) at \(t = t_n\), the coordinates \((x_n, y_n)\) of the center of \(\Phi_k\) is

\[
\begin{align*}
x_{n,k} &= x_n + (k - (K + 1)/2)\Delta r_C \cos \theta_n, \\
y_{n,k} &= y_n + (k - (K + 1)/2)\Delta r_C \sin \theta_n
\end{align*}
\]  
(9)

Then an exponential function is designed to compute \(e_{a,n}\) based on the relative velocity and distance between EV and \(a\). The meanings of \(e_{a,n,k}, \delta_{a,n,k}\), and \(\xi_{a,n,k}\) are shown in Fig. 2. \(rv_{a,n,k}\) is the relative velocity between \(a\) and \(\Phi_k\). \(d_{a,n,k}\) is the distance between the centers of \(a\) and \(\Phi_k\). With \(a\)’s position \((x_{a,n}, y_{a,n})\), velocity \(v_{a,n}\), and orientation \(\theta_{a,n}\), \(e_{a,n}\) is computed as

\[
\begin{align*}
d_{a,n,k} &= \sqrt{(x_{a,n} - x_{n,k})^2 + (y_{a,n} - y_{n,k})^2}; \\
\phi_{a,n,k} &= \arccos \left(\frac{x_{a,n} - x_{n,k}}{d_{a,n,k}}\right); \\
\delta_{a,n,k} &= \phi_{a,n,k} - \theta_n; \\
\xi_{a,n,k} &= \phi_{a,n,k} - \theta_{a,n}; \\
rv_{a,n,k} &= v_{a,n} \cos \xi_{a,n,k} - v_n \cos \delta_{a,n,k}; \\
\eta_{a,n} &= \begin{cases} 
1, & \text{if } a \text{ is currently in the same lane} \\
0, & \text{otherwise}; 
\end{cases} \\
e_{a,n} &= e_{a,n} \sum_{k=1}^{K} \exp(-0.1 \times rv_{a,n,k})/d_{a,n,k}
\end{align*}
\]  
(10)

where \(\varepsilon_{a,n} = 1\) means it is necessary to consider the collision risk of the \(n\)-th segment.

With \(J_{path}(p)\) and \(J_{drive}(p, a)\), the objective function for trajectory optimization is set to be

\[
J(p, a) = J_{path}(p) + J_{drive}(p, a)
\]  
(11)

which computes the comprehensive performance of a trajectory on its efficiency, safety, economy, and comfort. Since the trajectory curve is optimized in set \(\{I_n: g = 1, 2, \ldots, G\}\), the trajectory planning problem is formulated as the following NLP model,

\[
\begin{align*}
\Pi_1: & \quad \min_{p_a} J(p, a), \quad p \in P, \ a \in \mathbb{R}^N \\
\text{s.t.} & \quad h_j(a) \geq 0, \quad j = 1, 2, \ldots, 2N
\end{align*}
\]  
(12)

where \(h_{2n-1}(a) = a_n - C_{a_n}^l\), \(h_{2n}(a) = C_{a_n}^u - a_n\) \((n = 1, 2, \ldots, N)\). \(C_{a_n}^l\) and \(C_{a_n}^u\) are the lower and upper bounds of the acceleration.

From Eq. (12), two difficulties exist for solving \(\Pi_1\): (1) The solution space is large. The length of \(a\) is the number of trajectory segments (i.e., \(N\)), thus the size of the solution space grows exponentially with \(N\) increasing. (2) The computing burden of trajectory evaluation is heavy. The objective function \(J(p, a)\) involves computationally-expensive non-linear costs (e.g., \(C_{Coll}\)).

4 Trajectory Planning Based on OO

In order to solve the trajectory optimization model \(\Pi_1\) in real time, we develop a hybrid optimization algorithm which combines the idea of OO and a traditional intelligent optimization algorithm.

4.1 Idea of OO

OO was originally proposed by Ho et al.\(^{(18)}\) to deal with the optimization of Discrete Event Dynamic Systems (DEDS), whose evaluation is often quite time-consuming. Because of OO’s advantage on efficiently solving complex optimization problems, the idea of OO is applied here. OO has two basic ideas, which are illustrated as follows.

**Idea 1:** Use “Ordinal Comparison” instead of “Cardinal Comparison”. “Cardinal Comparison” (OC) means comparing unknown variables based on their accurate estimations, while “Ordinal Comparison” (OC) directly judges the order of variables in value (accurate estimations are unnecessary). Suppose that we compare two variables \(r_A\) and \(r_B\) \((r_A < r_B)\) but can only acquire their noisy observations \(X_A\) and \(X_B\). Then we can compare the means of \(N\) observations of \(r_A\) and \(r_B\), denoted by \(\overline{X}_A(N)\) and \(\overline{X}_B(N)\), respectively. For any number \(\varepsilon > 0\),

\[
\lim_{N \to +\infty} P\{|\overline{X}_A(N) - r_A| < \varepsilon\} =
\]
In addition, OC converges exponentially with respect to probability. In other words, noise is tolerable in OC.

Obviously, OC has lighter computation burden than CC, but we may arrive at $\mathbb{X}_A(N) > \mathbb{X}_B(N)$ with OC. We can compute the probability of making such incorrect judgment:

$$P\{\mathbb{X}_A(N) > \mathbb{X}_B(N)\} = \frac{1}{\sqrt{2\pi}} \int_l^{+\infty} \exp(-x^2/2)dx$$

(13)

where $l = (r_A - r_B)\sqrt{N/(\sigma_A^2 + \sigma_B^2)}$. $P\{\mathbb{X}_A(N) > \mathbb{X}_B(N)\}$ tends to be 0 when $N$ increases, which means that, as long as $N$ is large enough, we can pick the smaller one from $r_A$ and $r_B$ correctly with a high enough probability. In other words, noise is tolerable in OC. In addition, OC converges exponentially with respect to $N$, while for CC, “the limit $\lim_{N \to +\infty} \mathbb{X}_A(N) = r_A$, $\lim_{N \to +\infty} \mathbb{X}_B(N) = r_B$ converges with rate $O(1/\sqrt{N})$”[19], thus OC converges faster than CC.

Idea 2: Consider goal softening, i.e., be willing to settle for the “Good Enough” instead of insisting on getting the “Best”. To find a good design of the optimization variable $\theta$ in its search space $\Theta$, let us compare $u$ samples of $\theta$ from $\Theta$. Suppose that $|\cdot|$ represents the number of elements in the set, $|\Theta| = T$, and the $u$ samples are stored in set $U$. If the goal is to get the best $\theta$ in $\Theta$, i.e., $\theta_{opt}$, the probability that $\theta_{opt}$ is included in $U$ is $P\{\theta_{opt} \in U\} = u/T$. But if we regard the top-$g$ ($g > 1$) $\theta$s in $\Theta$ as good enough and soften the goal to getting a good enough $\theta$, the probability that $U$ includes at least 1 good enough $\theta$ is $P\{|U \cap G| \geq 1\} = 1 - C_T^g / C_T^U$, where $G$ is the set of good enough $\theta$s. Since $P\{|U \cap G| \geq 1\}$ converges to 1 at an exponential rate when $g$ increases[20], goal softening reduces the problem difficulty significantly.

Above two ideas are combined to solve $\Pi_1$. We could evaluate the feasible solutions of $\Pi_1$ in a biased but computationally-easier way, and then apply OC to determine the final solution. This avoids calculating time-consuming accurate trajectory evaluations. Good enough solutions are still ensured by controlling the level of biased evaluations.

4.2 Algorithm framework

To apply OO, $\Pi_1$ is first rewritten as $\Pi_2$ where the curve index $g$ and acceleration vector $a$ are optimized,

$$\Pi_2: \min_{g, a} J_1(g) + J_2(g, a), \quad g \in I, a \in \mathbb{R}^N$$

s.t., $h_j(a) \geq 0, \quad j = 1, 2, \ldots, 2N$

(14)

where $I = \{1, 2, \ldots, G\}$, $J_1(g) = J_{pab}(p_g)$, $J_2(g, a) = J_{\text{drive}}(p_g, a)$. $\Pi_2$ can be written in a two-layer form as $\min\{\min_{a} (J_1(g) + J_2(g, a))\}$, which can be further decomposed into an inner-layer problem $\Omega_g$ and an outer-layer problem $\Pi_3$.

$$\Omega_g: \min_{a} J_2(g, a), \quad a \in \mathbb{R}^N$$

s.t., $h_j(a) \geq 0, \quad j = 1, 2, \ldots, 2N$

$$\Pi_3: \min_{g} J_{o}(g), \quad g \in I$$

(15)

(16)

where $J_{o}(g) = J_1(g) + \min_{a} J_2(g, a)$ is computed by solving $\Omega_g$.

In $\Omega_g$, $a$ is optimized on the given curve $\Gamma_g$. Thus $J_{o}(g)$ is the performance evaluation of the best trajectory on curve $\Gamma_g$. Let us regard $J_{o}(g)$ as the evaluation of the candidate curve $\Gamma_g$, then solving $\Pi_3$ means picking the best curve from $\{\Gamma_g: g \in I\}$ by comparing $J_{o}(g)$.

Since only the index of the best curve is cared about in $\Pi_3$, OO is introduced to solve $\Pi_3$. We compute the rough (biased but computationally-easier) evaluation $\hat{J}_{o}(g)$ of each candidate curve $\hat{\Gamma}_g$ by roughly solving $\Omega_g$. Then instead of $J_{o}(g)$, we compare $\hat{J}_{o}(g)$ to select the best curve, so that $\Pi_1$ can be solved very efficiently if we settle for getting a good enough curve. In the rest of this paper, $\hat{J}_{o}(g)$ is denoted by “accurate evaluation” to distinguish $J_{o}(g)$ from $\hat{J}_{o}(g)$.

The inner-layer problem $\Omega_g$ has a highly-nonlinear objective function but involves only continuous optimization variables and linear constraints. As a representative of evolution algorithms which could work without computing the gradient, Differential Evolution (DE)[21] is applied to solve $\Omega_g$. Based on the two-layer optimization framework in Ref. [22], we propose OODE to solve the trajectory optimization model. The two-step framework of OODE is given in Fig. 3. In Step 1, for each candidate curve $\Gamma_g$, $\hat{J}_1(g)$ is directly computed with $p_g$, and $\hat{\Omega}_g$ is roughly solved by DE, so that $\hat{J}_{o}(g)$ is computed. Then the best curve in $\{\hat{\Gamma}_g: g \in I\}$, denoted by $\hat{\Gamma}_g$, is determined by comparing $\hat{J}_{o}(g)$. In Step 2, $\Omega_g$ is solved again accurately to compute the optimal acceleration vector $a_{\hat{g}}$. Finally, $(p_{\hat{g}}, a_{\hat{g}})$ is returned as the solution.

4.3 Algorithm implementation

For curve $\hat{\Gamma}_g$, OODE computes the rough evaluation $\hat{J}_{o}(g)$ by solving $\Omega_g$ roughly. “Roughly” is because the resolution of the vehicle trajectory model is decreased. The trajectory curve is divided more crudely, i.e., the number of trajectory segments takes $N_{c} (N_{c} < N)$
Instead of $N$. As a result, the problem size of $\Omega_g$ is decreased. Moreover, EV’s body is approximated by $K_c (K_c < K)$ circles rather than $K$ ones, so that the calculation of the trajectory’s collision risk is simplified. Then the time consumed for evaluating a design in $\Omega_g$ is reduced. With these two techniques, $\Omega_g$ can be solved very fast.

DE/rand/1/bin is applied to solve $\Omega_g$. “rand/1/bin” means during the creation of mutant individuals, the base individual is “randomly” chosen from the population, “1” individual difference is added to the base individual, and the number of parameters donated by the mutant individual follows a “binomial” distribution\[^{21}\].

The rough evaluation $J_0(g)$ is computed as

$$J_0(g) = J_1(g) + J_2(g, a_{\text{opt}})$$

where $a_{\text{opt}}$ is the rough solution. The index of the “best” curve determined by comparing rough curve evaluations is

$$g = \min_g J_0(g)$$

In Step 2, the inner-layer problem corresponding to the selected “best” curve, i.e., $\Omega_g$, is “accurately” solved. “Accurately” means that, the original trajectory model resolution is used, i.e., the trajectory curve is still divided into $N$ segments and the vehicle body is still approximated by $K$ circles.

The algorithm applied here is DE/best/1/bin, whose differences from DE/rand/1/bin are: (1) The population is initialized according to the rough solution obtained by solving the same inner-layer problem in Step 1, which can speed up the convergence. (2) At each iteration, the mutant individual is created based on the best found individual, so that the computation resource is focused on local searching.

With the solution $a_g$ of $\Omega_g$, the solution of the NLP model $\Pi_1$ returned by OODE is $(p_g, a_g)$.

### 5 Simulation Result

In this section, the planning result and performance of the proposed method are demonstrated and analyzed. All the simulations are executed in Visual Studio 2010 using an Intel(R) Core(TM) i5 CPU M 480@2.67 GHz with RAM 2.0 GB. A typical traffic scenario on straight roads, including three obstacles (vehicles) $\alpha, \beta,$ and $\gamma$, is considered (see Fig. 4). All obstacles are assumed to move along lane centerlines at constant speeds.

The initial state of the traffic scenario is denoted by a set of scenario parameters. $\kappa_0, \theta_0, y_0, v_0,$ and $a_0$ are EV’s initial curvature, orientation, lateral position, velocity, and acceleration, respectively. $x_\alpha, x_\beta,$ and $x_\gamma$ are the longitudinal distance between $\alpha, \beta,$ and $\gamma$, respectively. $v_\alpha, v_\beta,$ and $v_\gamma$ are the velocity of $\alpha, \beta,$ and $\gamma$, respectively. A problem instance of trajectory planning can be represented by the scenario parameters. We simulate 4 specific scenarios ($S_1, S_2, S_3,$ and $S_4$), and introduce a test set of 2000 scenarios, whose scenario parameters take random values. Let SR denote such a random scenario. The parameters of $S_1, S_2, S_3, S_4,$ and SR are given in Table 3, where $U(a, b)$ represents the uniform distribution on interval $[a, b]$.

The parameters of OODE are listed in Table 4. $I_c, N_c, NP_c, F_c,$ and $CR_c$ are the number of iterations, number of optimization variables, population size, differential weight, and crossover probability of DE/rand/1/bin, respectively. $I, N, NP, F,$ and $CR$ are the number of iterations, number of optimization variables, population size, differential weight, and crossover probability of DE/best/1/bin, respectively. The weights of path costs and driving costs in the objective function are mainly
selected by experience. The weights of critical costs (curve length, collision risk) are firstly determined. Then the rest are selected via simulation.

### 5.1 Trajectory planning result

The proposed method is applied to plan trajectories in Scenarios S1 and S2. The planning results are shown in Fig. 5. Figures 5a and 5b demonstrate the locations of EV, α, β, and γ at various time points in Scenarios S1 and S2, respectively. The trajectory curve is denoted by a dashed curve with a circle and a star representing the trajectory start and end points, respectively. EV and obstacles are represented by gray and white rectangles, respectively. Figures 5c and 5d show the acceleration profiles in Scenarios S1 and S2, respectively. The 25 blocks represent the accelerations in 25 trajectory segments.

In both S1 and S2, the preceding vehicle β moves slower than EV. In Scenario S1, since the left and right adjacent lanes are both occupied, EV keeps driving

---

**Table 3** Scenario parameters of S1, S2, S3, S4, and SR.

| Scenario | y₀ (feet) | θ₀ (rad) | κ₀ (1/feet) | v₀ (feet/s) | a₀ (feet/s²) | x₀ (feet) | v₀ (feet/s) | x₀ (feet) | v₀ (feet/s) | x₀ (feet) | v₀ (feet/s) |
|----------|-----------|----------|-------------|-------------|-------------|-----------|-------------|-----------|-------------|-----------|-------------|
| S1       | −2        | 0.01     | 0.01        | 35          | −5          | 30        | 25          | 40        | 30          | 20        | 20          |
| S2       | 1         | −0.1     | 0           | 30          | 0           | 80        | 40          | 50        | 20          | 10        | 40          |
| S3       | 5         | 0.3      | 0.01        | 40          | 4           | −50       | 30          | 80        | 40          | −80       | 50          |
| S4       | −3        | −0.4     | 0.02        | 20          | −2          | 10        | 50          | 160       | 50          | −20       | 20          |
| SR       | U(−6.6, 6) | U(−0.5, 0.5) | U(0, 0.02) | U(0, 0.02) | U(40, 200) | U(10, 60) | U(−100, 100) | U(10, 60) | U(40, 200) | U(10, 60) |

Note: 1 feet = 0.3048 m.

**Table 4** Parameters of OODE.

| I₀ | N₀ | NP₀ | F₀ | CR₀ | G₀ | K₀ | ΔIC (feet) | RC (feet) | Lplan (feet) | dplan (feet) | wLen | wCurv | wACurv | wOffs |
|----|----|-----|----|-----|----|----|------------|-----------|-------------|-------------|------|-------|--------|-------|
| 80 | 5  | 10  | 0.85 | 0.95 | 63 | 1  | 3.0        | 3.0       | 65.0        | 12.0        | 0.2  | 0.05  | 3 × 10⁴ | 700   |

| I₀ | N₀ | NP₀ | F₀ | CR₀ | G₀ | K₀ | ΔIC (feet) | RC (feet) | Lplan (feet) | dplan (feet) | wLen | wCurv | wACurv | wOffs |
|----|----|-----|----|-----|----|----|------------|-----------|-------------|-------------|------|-------|--------|-------|
| 80 | 5  | 10  | 0.85 | 0.95 | 63 | 1  | 3.0        | 3.0       | 65.0        | 12.0        | 0.2  | 0.05  | 3 × 10⁴ | 700   |

Note: 1 feet = 0.3048 m.
in its current lane (Lane B) and slows down to keep distance from $\beta$. But in Scenario S2, Lane A is available for entering, so that the optimal trajectory for EV is to change its lane to Lane A. EV first controls its speed to adjust the space headway, and then accelerates to perform the lane change behavior.

In addition, our method is simulated in a receding horizon fashion. The replanning cycle is 0.3 s. Figure 6 gives an example, where EV overtakes a slower moving vehicle in the middle lane by performing a double lane change maneuver.

5.2 Algorithm performance analysis

The optimality of the trajectory computed by OODE is investigated. We apply OODE to solve the 2000 problem instances in the test set. In each experiment, the accurate evaluation of each candidate curve is computed, so that the real rank (in all $G$ candidate curves) of the “best” curve selected by comparing rough evaluations could be obtained. Then we can compute the percent of the experiments where the “best” curve truly ranks in the top $\xi$%, which is denoted by Top Rank Percent (TRP). The relationship between $\xi$ and TRP is shown in Fig. 7. OODE gets a top-5% trajectory curve with probability 0.95 and a top-10% curve with probability 0.97. We can say that, a good enough trajectory curve is ensured with a high enough probability.

The efficiency of OODE is compared with other two algorithms (DE and TRA). The three algorithms (DE, TRA, OODE) are implemented to solve the problem instances in Scenarios S1, S2, S3, S4, and the test set. As an intelligent optimization algorithm, DE redefines the feasible goal point for trajectory planning to be distributed continuously along the lane centerlines, and then directly solves $\Pi_1$ with DE/rand/1/bin\cite{21}. TRA solves the inner-layer problem corresponding to each candidate curve “accurately” with DE/rand/1/bin\cite{21}, and then returns the best found trajectory. In Table 5, “$J$ for S1, S2, S3, S4” represent the performance evaluations of the obtained solutions in Scenarios S1, S2, S3, S4, respectively. “$\bar{J}$ for SR” is the mean evaluation of the solutions of the problems in the test set. “$\bar{T}$” is the mean CPU time for solving the problems in the test set.

Because DE optimizes the curve parameter vector $p$ and the acceleration vector $a$ simultaneously in a continuous space, DE statistically returns a better solution than TRA and OODE (see the column “$J$ for SR”). But DE is sometimes trapped in local minimum (e.g., “$J$ for S4”) since DE searches for the solution in a high-dimensional, non-convex space.

The solution from OODE is slightly worse than TRA after introducing rough curve evaluations. But OODE selects the best curve from candidates by comparing rough curve evaluations. Since OC is robust against noise and converges significantly faster than CC, the best curve is determined very efficiently. As a result, OODE is obviously faster than DE and TRA. Table 6 further compares the time consumed for computing a rough and accurate curve evaluation, denoted by $T_0$. With the trajectory model resolution decreased ($N$ and $K$ are reduced to $N_c$ and $K_c$), $T_0$ is reduced significantly. OODE achieves much higher efficiency than TRA.

### Table 5 Algorithm performance of DE, TRA, and OODE.

| Algorithm | $J$ for S1 | $J$ for S2 | $J$ for S3 | $J$ for S4 | $\bar{J}$ for SR | $\bar{T}$ (s) |
|-----------|-----------|-----------|-----------|-----------|-----------------|-------------|
| DE        | 42.19     | 41.86     | 30.15     | 35.32     | 38.19           | 2.45        |
| TRA       | 42.27     | 44.39     | 30.30     | 32.14     | 40.36           | 11.80       |
| OODE      | 42.16     | 44.49     | 30.30     | 32.14     | 40.55           | 0.24        |
From Table 5 and Fig. 5, OODE returns a trajectory of time length 1.5 s in less than 250 ms. During the planning period, the largest estimation error of the obstacle position introduced by the constant speed assumption is \( 0.5 \times 12 \text{ feet/s}^2 \times (0.25 \text{ s})^2 = 0.375 \text{ feet} \), where 12 feet/s\(^2\) is the obstacle’s largest acceleration/deceleration. Consequently, this assumption is believed to be valid, and OODE meets the real-time demand.

6 Conclusion

This paper proposes an ordinal-optimization-based approach to solve trajectory planning for automated driving. The main contribution includes an NLP model for trajectory optimization and an optimization algorithm OODE. Compared with the existing approaches based on candidate curves which returns a suboptimal solution, we optimize the trajectory curve and acceleration profile simultaneously in an unified NLP model. OODE gets a good enough solution with a high probability by selecting the “best” curve from candidates based on rough curve evaluations. With OO introduced, OODE achieves high enough efficiency to work in real time.

However, it is still not clear how solving the inner-layer problem roughly results in the error in curve evaluation. Studying the relationship between the trajectory model resolution and the curve evaluation is helpful for understanding the principle of OODE. This will be researched in the future.

In order to apply the proposed approach in real systems, more experiments are required to improve the NLP model (e.g., revise the cost formulas, set appropriate cost weights). Moreover, introducing priori knowledge to assist trajectory optimization (e.g., initialize the acceleration vectors according to the traffic environment) can further speed up planning and improve the result.

References

[1] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner, Three decades of driver assistance systems: Review and future perspectives, IEEE Intelligent Transportation Systems Magazine, vol. 6, no. 4, pp. 6–22, 2014.

[2] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, N. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al., Autonomous driving in urban environments: Boss and the urban challenge, Journal of Field Robotics, vol. 25, no. 8, pp. 425–466, 2008.

[3] T. Nothdurft, P. Hecker, S. Ohl, F. Saust, M. Maurer, A. Reschka, and J. R. Böhmer, Stadtpilot: First fully autonomous test drives in urban traffic, in 2011 14th International IEEE Conference on Intelligent Transportation Systems, Washington DC, USA, 2011, pp. 919–924.

[4] A. Broggi, M. Buzzoni, S. Debattisti, P. Grisleri, M. C. Laghi, P. Medici, and P. Versari, Extensive tests of autonomous driving technologies, IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 3, pp. 1403–1415, 2013.

[5] L. Ma, J. Xue, K. Kawabata, J. Zhu, C. Ma, and N. Zheng, Efficient sampling based motion planning for on-road autonomous driving, IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 4, pp. 1961–1976, 2015.

[6] T. Hesse and T. Sattel, An approach to integrate vehicle dynamics in motion planning for advanced driver assistance systems, in Proceedings of the 2007 IEEE Intelligent Vehicles Symposium, Istanbul, Turkey, 2007, pp. 1240–1245.

[7] J. Hilgert, K. Hirsch, T. Bertram, and M. Hiller, Emergency path planning for autonomous vehicles using elastic band theory, in Proceedings of the 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Duisburg, Germany, 2003, pp. 1390–1395.

[8] Q. Gong, L. R. Lewis, and I. M. Ross, Pseudospectral motion planning for autonomous vehicles, Journal of Guidance, Control, and Dynamics, vol. 32, no. 3, pp. 1039–1045, 2009.

[9] S. J. Anderson, S. C. Peters, and T. E. Pilutti, Design and development of an optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios, in Robotics Research: The 14th International Symposium ISRR, C. Pradalier, R. Siegwart, and G. Hirzinger, eds. Springer, 2011, pp. 39–54.

[10] M. McNaughton, C. Urmson, J. M. Dolan, and J. W. Lee, Motion planning for autonomous driving with a conformal spatiotemporal lattice, in 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 2011, pp. 4889–4895.

[11] J. Ziegler and C. Stiller, Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios, in The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St Louis, MO, USA, 2009, pp. 1879–1884.

[12] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, Real-time motion planning with applications to autonomous urban driving, IEEE Transactions on Control Systems Technology, vol. 17, no. 5, pp. 1105–1118, 2009.

Table 6 Comparison of computing rough and accurate curve evaluations.

| Evaluation method | \( T_0 \) (ms) | \( N_c \) | \( K_c \) |
|-------------------|---------------|---------|---------|
| Accurate evaluation | 192.0         | 25      | 5       |
| Rough evaluation  | 1.1           | 5       | 1       |
[13] I. Papadimitriou and M. Tomizuka, Fast lane changing computations using polynomials, in Proceedings of the American Control Conference, Denver, CO, USA, 2003, pp. 48–53.

[14] S. Köhler, B. Schreiner, S. Ronalter, K. Doll, U. Brunsmann, and K. Zindler, Autonomous evasive maneuvers triggered by infrastructure-based detection of pedestrian intentions, in 2013 IEEE Intelligent Vehicles Symposium (IV), Gold Coast, Australia, 2013, pp. 519–526.

[15] M. Montemerlo, J. Becker, S. Bhat, H. Dhlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, et al., Junior: The stanford entry in the urban challenge, Journal of Field Robotics, vol. 25, no. 9, pp. 569–597, 2008.

[16] S. Glaser, B. Vanholme, S. Mammar, D. Gruyer, and L. Nouvelière, Maneuver based trajectory planning for highly autonomous vehicles on real road with traffic and driver interaction, IEEE Transactions on Intelligent Transportation Systems, vol. 11, no. 3, pp. 589–606, 2010.

A. Kelly and B. Nagy, Reactive nonholonomic trajectory generation via parametric optimal control, The International Journal of Robotics Research, vol. 22, nos. 7&8, pp. 583–601, 2003.

Y. Ho, Q. Zhao, and Q. Jia, Ordinal Optimization: Soft Optimization for Hard Problems. Springer, 2007.

L. Dai, Convergence properties of ordinal comparison in the simulation of discrete event dynamic systems, Journal of Optimization Theory and Applications, vol. 91, no. 2, pp. 363–388, 1996.

H. Lee, T. W. E. Lau, and Y. C. Ho, Explanation of goal softening in ordinal optimization, IEEE Transactions on Automatic Control, vol. 44, no. 1, pp. 94–99, 1999.

K. V. Price, R. M. Storn, and J. A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization. Springer, 2005.

Dexian Huang received the BS degree from China University of Petroleum (Huadong), Dongying, China, in 1982, the MS degree in industrial automation from China University of Petroleum (Beijing), China, in 1988, and the PhD degree in control theory and control engineering from Tsinghua University, China, in 2000. From 1992 to 1995, he was an associate professor with Dept. of Automation, China University of Petroleum (Huadong). From 1995 to 2000, he was a professor, with Dept. of Automation, China University of Petroleum (Huadong). Since 2000, he has been a professor with Dept. of Automation, Tsinghua University. His research interests include the modeling, simulation, control, and real-time optimization of process industry.

Jingchun Wang received the BS degree and the MS and PhD degrees in control theory and engineering from Tsinghua University, China, in 1992, 1994, and 1997, respectively. Since 1999, he has been an associate professor of Department of Automation, Tsinghua University. His research interests include process control theory and its applications, such as multivariable predictive control, CIMS related theory and system design, supply chain, and real time database system for process control.