

Intelligent computing budget allocation for on-road trajectory planning based on candidate curves*

Xiao-xin FU^{†1}, Yong-heng JIANG^{†‡1}, De-xian HUANG¹, Jing-chun WANG¹, Kai-sheng HUANG²

(¹Department of Automation, Tsinghua University, Beijing 100084, China)

(²Department of Automotive Engineering, Tsinghua University, Beijing 100084, China)

[†]E-mail: fuxx10@mails.tsinghua.edu.cn; jiangyh@tsinghua.edu.cn

Received Aug. 18, 2015; Revision accepted Jan. 29, 2016; Crosschecked May 18, 2016

Abstract: In this paper, on-road trajectory planning is solved by introducing intelligent computing budget allocation (ICBA) into a candidate-curve-based planning algorithm, namely, ordinal-optimization-based differential evolution (OODE). The proposed algorithm is named IOODE with ‘I’ representing ICBA. OODE plans the trajectory in two parts: trajectory curve and acceleration profile. The best trajectory curve is picked from a set of candidate curves, where each curve is evaluated by solving a subproblem with the differential evolution (DE) algorithm. The more iterations DE performs, the more accurate the evaluation will become. Thus, we intelligently allocate the iterations to individual curves so as to reduce the total number of iterations performed. Meanwhile, the selected best curve is ensured to be one of the truly top curves with a high enough probability. Simulation results show that IOODE is 20% faster than OODE while maintaining the same performance in terms of solution quality. The computing budget allocation framework presented in this paper can also be used to enhance the efficiency of other candidate-curve-based planning methods.

Key words: Intelligent computing budget allocation, Trajectory planning, On-road planning, Intelligent vehicles, Ordinal optimization

<http://dx.doi.org/10.1631/FITEE.1500269>

CLC number: TP242.6

1 Introduction

1.1 Background


Intelligent vehicles have become a hot topic all around the world because of their great potential in increasing road utilization, transportation efficiency, and driving safety. For the past three decades, the research of this field focused mainly on two applications: driver assistance systems (DASs) and driverless vehicles. DASs, such as adaptive cruise control (Bengler *et al.*, 2014), are installed on manned vehi-

cles to enhance driving experience. Driverless vehicles, such as the vehicles competing in the DARPA Urban Challenge (Montemerlo *et al.*, 2008; Urmson *et al.*, 2008) and the autonomous vehicle A1 (Chu *et al.*, 2012), navigate autonomously in complex environments (e.g., highways) with only on-board sensors and computers. For either of these applications, trajectory planning is a fundamental and important technology as it can be stated as “computing a sequence of control values or feasible movement states for the vehicle to maneuver among obstacles from an initial state toward a desired terminal state, taking into account the vehicle’s kinematic and dynamic model” (Ma *et al.*, 2015).

This paper focuses on on-road trajectory planning, which deals with the generation of

[‡] Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 61273039)

 ORCID: Yong-heng JIANG, <http://orcid.org/0000-0002-9551-9846>

© Zhejiang University and Springer-Verlag Berlin Heidelberg 2016

trajectories in the urban road environment. This problem is known to be computationally challenging, since a simple version of it, the piano mover's problem, has proven to be PSPACE-hard (Reif, 1979). However, there are more difficulties. To obtain the best driving experience, the trajectory efficiency, safety, comfort, and economy all need to be carefully modeled and optimized, which inevitably introduces computationally expensive trajectory performance evaluation. In addition, the driving environment changes very fast due to the existence of moving obstacles, and thus the planning result needs to be updated in real time, which sets a high requirement for the planning speed. As a consequence, planning efficiency becomes the most significant concern in this problem.

1.2 Related work

In the field of on-road vehicle trajectory planning, researchers have proposed a lot of methods, many of which were originally designed for robots. We present an overview of the popular approaches as follows. Their advantages and disadvantages will be discussed. The most concerned criterion, their efficiency (the quality of the returned solution and the real-time performance), will be commented on.

Potential field methods (Hilgert *et al.*, 2003; Gehrig and Stein, 2007) can easily produce a collision-free path in real time, but they suffer from the local optimum problem. Although the distance between the vehicle and obstacles can be adjusted by modifying algorithm parameters, it is hard to model and optimize other complex trajectory costs with potentials (e.g., jerk and collision risk).

State lattice methods (Ziegler and Stiller, 2009; McNaughton *et al.*, 2011) apply graph-based search to produce the optimal path through a search graph of primitive trajectories (the graph looks like a lattice). With the addition of the vehicle's velocity component to the state space, these methods allow the search algorithm to explore both spatial and temporal dimensions. However, due to the exponential growth of the search space with respect to the resolution of discretization, state lattice methods are not efficient enough for dynamic traffic scenarios.

Sampling-based planning methods save large amounts of computation by building the tree or graph of short trajectories via repeated sampling

in the obstacle-free state space instead of using an explicit environment representation. Like state lattice methods, the optimality of the solution depends on the resolution of sampling. Although sampling-based methods, such as rapidly-exploring random tree (RRT) (Kuwata *et al.*, 2009; Ma *et al.*, 2015), have been implemented to work in real time on powerful computers, they cannot run on simple microprocessors (Glaser *et al.*, 2010), which are more likely and feasible to be applied on commercial vehicles in the near future and will be used in this paper for algorithm simulation.

Parametric planning methods model the trajectories with parametric expressions, e.g., polynomials (Papadimitriou and Tomizuka, 2003) and clothoids (Köhler *et al.*, 2013), where the unknown parameters are optimized to compute the trajectory solution. These methods are applied mainly in situations with high real-time requirements (such as collision avoidance) because of the benefit of their simplicity. However, since the kinematic constraints are not considered, accurate execution of the produced trajectory would be not easy.

To achieve a trade-off between planning speed and the optimality of the solution, some researchers developed candidate-curve-based planning methods (Montemerlo *et al.*, 2008; Urmson *et al.*, 2008; Glaser *et al.*, 2010; Chu *et al.*, 2012). A set of candidate curves which satisfy vehicle constraints is first generated. Then the best curve is picked from the candidates, and the velocity profile which characterizes how the vehicle velocity changes along that curve is computed. These methods have been successfully applied on driverless vehicles. However, since they either use a linear velocity profile or directly specify the target driving speed, a careful optimization of the velocity profile is lacking. Also, because the best curve and the velocity profile are computed separately, the obtained trajectory is suboptimal.

1.3 Motivation

For the planning methods based on candidate curves, the curves should be distributed densely enough to cover all possible maneuvers on roads. To obtain a globally good trajectory, it is also important to consider the variation of the velocity profile when comparing candidate curves. Thus, the key challenge is how to efficiently select a satisfactory curve from

the candidates.

In our previous work (Fu *et al.*, 2015), based on candidate curves, the planning problem was formulated as a non-linear programming (NLP) model, which optimizes the trajectory curve and the acceleration profile simultaneously (the velocity profile is the integral of the acceleration profile). To solve the NLP model in real time, we developed a hybrid intelligent optimization algorithm named ‘ordinal-optimization-based differential evolution’ (OODE). OODE compares the rough (biased but computationally easy) performance evaluations of candidate curves, from which a good curve is selected in a short time. Then the acceleration profile was optimized with that curve. OODE can generate a trajectory of time length 1.5 s in 250 ms. The solution was ensured to be globally good enough in the sense of probability.

In Fu *et al.* (2015), the rough evaluation of each candidate curve is the optimal value of a corresponding objective function, optimized by the differential evolution (DE) algorithm. The more iterations DE performs, the more accurate the evaluation will become. However, since an equal number of iterations is conducted for each curve while we need only to pick the best one, a large amount of computing time is spent on obtaining better evaluations of unconcerned curves. If a larger portion of the computing budget (measured by the total number of iterations for all curves) is allocated to those curves that are critical to identifying the best, the efficiency of OODE can be improved.

The ranking-and-selection (R&S) procedures, which are intended to select the best of a finite set of alternatives, have been extensively researched in the past three decades (Branke *et al.*, 2007; Chen and Lee, 2010; Chen *et al.*, 2015). The best is determined with respect to the largest/smallest mean, but the mean must be inferred via statistical sampling (Bechhofer *et al.*, 1995). There are two points of view on defining the evidence for correct selection, which motivate two categories of approaches to the selection problem: (1) frequentist approaches, e.g., the indifference zone (IZ) (Kim and Nelson, 2001); (2) Bayesian approaches, e.g., the expected value of information procedure (VIP) (Chick and Inoue, 2001) and the optimal computing budget allocation (OCBA) (Chen *et al.*, 2000). The IZ approaches

typically allocate samples to provide a guaranteed lower bound for the frequentist probability of correct selection. The VIP and OCBA approaches describe the evidence for correct selection with the Bayesian posterior distribution of the unknown mean performance of each alternative, and allocate the further samples to maximize that evidence. A variant of OCBA also proposes a different perspective: minimize the total computation cost with a desired level of selection quality achieved (Chen and Yüesan, 2005). The above approaches achieve higher selection efficiency than the plain equal allocation schema.

Note that the procedure of identifying the best curve based on rough curve evaluation in OODE is similar to the R&S procedures. The curve evaluation is random (as DE is a random algorithm) and converges to its true value as the number of iterations increases. Thus, the computing budget can also be allocated sequentially. Inspired by the Bayesian approaches for R&S, we can define a similar measurement of selection quality, based on the posterior distribution of each unknown curve evaluation. Then we allocate the budget intelligently according to that measurement. The computing cost can be reduced. In this paper, the new algorithm is named ‘IOODE’, short for intelligent OODE.

2 Candidate-curve-based trajectory planning

2.1 Trajectory optimization model

The vehicle trajectory is modeled in two parts: the trajectory curve and the acceleration profile. A trajectory curve can be determined by the curve’s total length s_f and a function $\kappa = u(s)$ from driving distance s ($0 \leq s \leq s_f$) to curve curvature κ . Here, $u(s)$ is implemented as a polynomial function. We combine the polynomial coefficients of $u(s)$ and s_f to form a vector of size ζ , which is defined to be the trajectory curve’s parameter vector \mathbf{p} ($\mathbf{p} \in \mathbb{R}^\zeta$). Then with \mathbf{p} , we can compute the vehicle’s pose vector \mathbf{s} ($\mathbf{s} = [x \ y \ \theta \ \kappa]^T$) at any point of the curve, which consists of position (x, y) , orientation (θ) , and curvature (κ) . The trajectory curve is divided into N segments of equal length, where the vehicle is assumed to move at constant acceleration in each segment.

The accelerations a_1, a_2, \dots, a_N in N segments are stored in the acceleration vector \mathbf{a} ($\mathbf{a} \in \mathbb{R}^N$), which denotes the acceleration profile. Therefore, with given (\mathbf{p}, \mathbf{a}) , a trajectory is determined.

To evaluate the performance of a trajectory, the vehicle's body is represented by K circles $\Phi_1, \Phi_2, \dots, \Phi_K$ (Fig. 1), which are used to calculate the collision risk between the vehicle and obstacles. The trajectory's static performance evaluation $J_{\text{path}}(\mathbf{p})$ and dynamic performance evaluation $J_{\text{drive}}(\mathbf{p}, \mathbf{a})$ are computed as the weighted sum of path costs and driving costs, respectively:

$$\begin{aligned} J_{\text{path}}(\mathbf{p}) &= w_{\text{Leng}} \cdot C_{\text{Leng}} + w_{\text{pCurv}} \cdot C_{\text{pCurv}} \\ &+ w_{\text{dCurv}} \cdot C_{\text{dCurv}} + w_{\text{Offs}} \cdot C_{\text{Offs}}, \quad (1) \\ J_{\text{drive}}(\mathbf{p}, \mathbf{a}) &= w_{\text{Time}} \cdot C_{\text{Time}} + w_{\text{pAcce}} \cdot C_{\text{pAcce}} \\ &+ w_{\text{dAcce}} \cdot C_{\text{dAcce}} + w_{\text{Spd}} \cdot C_{\text{Spd}} + w_{\text{Coll}} \cdot C_{\text{Coll}}, \quad (2) \end{aligned}$$

where $w_{\text{Leng}}, w_{\text{pCurv}}, w_{\text{dCurv}}, w_{\text{Offs}}$ are the weights of path costs $C_{\text{Leng}}, C_{\text{pCurv}}, C_{\text{dCurv}}, C_{\text{Offs}}$, respectively, and $w_{\text{Time}}, w_{\text{pAcce}}, w_{\text{dAcce}}, w_{\text{Spd}}, w_{\text{Coll}}$ are the weights of driving costs $C_{\text{Time}}, C_{\text{pAcce}}, C_{\text{dAcce}}, C_{\text{Spd}}, C_{\text{Coll}}$, respectively. The computations of path costs and driving costs are given in Table 1, where $\Delta s_f = s_f/N$ is the length of each trajectory curve segment. For the n th trajectory segment, the collision risk is

$$e_n = \sum_{d=1}^D \sum_{k=1}^K \rho(v_o, d_o),$$

where $\rho(\cdot)$ is an exponential function, v_o and d_o are the relative velocity and distance between the k th circle Φ_k (which represents the vehicle body, $k = 1, 2, \dots, K$, Fig. 1) and the d th obstacle, respectively, and D is the total number of obstacles.

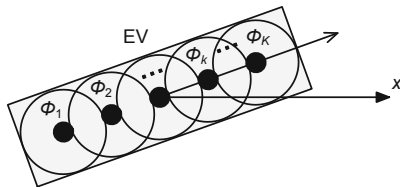


Fig. 1 Representation of a vehicle's body

From Table 1, trajectory safety is evaluated with the illegal velocity risk (C_{Spd}) and the collision risk (C_{Coll}). Trajectory efficiency is evaluated with the

Table 1 Path costs and driving costs

Cost	Computing formula	Interpretation
C_{Leng}	s_f	Trajectory curve length
C_{pCurv}	$\frac{1}{2} \int_0^{s_f} \kappa(s)^2 ds$	$\kappa(s)$ is the trajectory curve curvature
C_{dCurv}	$\frac{1}{2} \int_0^{s_f} \dot{\kappa}(s)^2 ds$	$\dot{\kappa}(s)$ is the curve curvature derivative
C_{Offs}	$\frac{1}{2} \sum_{n=1}^N \chi_n^2 \cdot \Delta s_f$	χ_n is the curvature difference between trajectory curve and lane centerline
C_{Time}	t_N	Time consumed for trajectory execution
C_{pAcce}	$\sum_{n=1}^N a_n^2 \cdot \Delta s_f$	a_n is the acceleration
C_{dAcce}	$\sum_{n=1}^N \Delta a_n^2 \cdot \Delta s_f$	Δa_n is the acceleration increment
C_{Spd}	$\sum_{n=1}^N \eta_n \cdot \Delta s_f$	η_n is the illegal velocity risk
C_{Coll}	$\sum_{n=1}^N e_n \cdot \Delta s_f$	e_n is the collision risk

$\Delta s_f = s_f/N$

trajectory curve length (C_{Leng}) and the execution time (C_{Time}). Trajectory economy is evaluated with the integral of the squared curvature (C_{pCurv}) and the sum of the squared accelerations (C_{pAcce}). Trajectory comfort (including jerk) is evaluated with the integral of the squared curvature derivative (C_{dCurv}) and the sum of the squared acceleration increments (C_{dAcce}). Therefore, $J_{\text{path}}(\mathbf{p})$ and $J_{\text{drive}}(\mathbf{p}, \mathbf{a})$ together describe the comprehensive performance of the vehicle trajectory determined by (\mathbf{p}, \mathbf{a}) .

However, since $\mathbf{p} \in \mathbb{R}^\zeta$, $\mathbf{a} \in \mathbb{R}^N$, the feasible region of the trajectory solution grows exponentially with the increase of ζ and N . To narrow the search, a set of candidate trajectory curves $\{\Gamma_1, \Gamma_2, \dots, \Gamma_G\}$ is generated, and the best curve is picked from $\{\Gamma_1, \Gamma_2, \dots, \Gamma_G\}$. Each curve Γ_g ($g \in \mathbb{I}$, $\mathbb{I} = \{1, 2, \dots, G\}$) is designed to lead the vehicle to a predefined goal pose \mathbf{s}_g . The parameter vector of Γ_g , denoted by \mathbf{p}_g , is obtained by optimizing the static performance evaluation $J_{\text{path}}(\mathbf{p})$:

$$\begin{aligned} \Psi_g : \quad & \min_{\mathbf{p}} J_{\text{path}}(\mathbf{p}), \quad \mathbf{p} \in \mathbb{R}^\zeta \\ \text{s.t.} \quad & f_i(\mathbf{s}_g, \mathbf{p}) = 0 \quad (i = 1, 2, 3, 4), \quad l(\mathbf{p}) > 0, \quad (3) \end{aligned}$$

where $f_i(\mathbf{s}_g, \mathbf{p}) = 0$ is defined for each component of the pose vector for $i = 1, 2, 3, 4$, to guarantee that the curve determined by \mathbf{p} is ended with \mathbf{s}_g , and

$l(\mathbf{p}) = s_f > 0$ is introduced to bound the element s_f of \mathbf{p} .

Now with G candidate trajectory curves, the vehicle trajectory is planned by optimizing the trajectory curve index g and the acceleration vector \mathbf{a} in the following NLP model:

$$\begin{aligned} \Pi_1 : \quad & \min_{g, \mathbf{a}} [J_1(g) + J_2(g, \mathbf{a})], \quad g \in \mathbb{I}, \mathbf{a} \in \mathbb{R}^N, \\ \text{s.t.} \quad & h_j(\mathbf{a}) \geq 0, \quad j = 1, 2, \dots, 2N, \end{aligned} \tag{4}$$

where $h_j(\mathbf{a}) \geq 0$ ($j = 1, 2, \dots, 2N$) are the boundary constraints for the elements in \mathbf{a} . Since $J_1(g) = J_{\text{path}}(\mathbf{p}_g)$ and $J_2(g, \mathbf{a}) = J_{\text{drive}}(\mathbf{p}_g, \mathbf{a})$, $[J_1(g) + J_2(g, \mathbf{a})]$ is the performance evaluation of the trajectory determined by $(\mathbf{p}_g, \mathbf{a})$. For details about the trajectory optimization model, please refer to Fu et al. (2015).

2.2 Framework of OODE

The search space can be much reduced by introducing candidate curves, but it is still challenging to solve Π_1 , because trajectory evaluation is quite time-consuming ($J_2(g, \mathbf{a})$ involves complex non-linear costs, e.g., C_{Coll}) while Π_1 needs to be solved in real time. Therefore, OODE is proposed to solve Π_1 .

First, Π_1 is rewritten in a two-layer form as $\min_g [\min_{\mathbf{a}} (J_1(g) + J_2(g, \mathbf{a}))]$, which can be further decomposed into an inner-layer problem Ω_g and an outer-layer problem Π_2 :

$$\begin{aligned} \Omega_g : \quad & \min_{\mathbf{a}} J_2(g, \mathbf{a}), \quad \mathbf{a} \in \mathbb{R}^N \\ \text{s.t.} \quad & h_j(\mathbf{a}) \geq 0, \quad j = 1, 2, \dots, 2N, \end{aligned} \tag{5}$$

$$\Pi_2 : \quad \min_g J_O(g), \quad g \in \mathbb{I}, \tag{6}$$

where $J_O(g) = J_1(g) + \min_{\mathbf{a}} J_2(g, \mathbf{a})$ depends on the optimal objective function value of Ω_g . Since $J_O(g)$ represents the performance evaluation of the optimal trajectory on curve Γ_g (with given \mathbf{p}_g , only \mathbf{a} is optimized), $J_O(g)$ can be seen as the evaluation of Γ_g . Then solving Π_2 means picking the best curve from $\{\Gamma_g: g \in \mathbb{I}\}$ by comparing evaluation $J_O(g)$.

OODE is designed based on ordinal optimization (OO), which has two tenets: (1) ‘order’ is easier than ‘value’; (2) settle for the ‘good enough’ instead of insisting on obtaining the ‘best’ (Ho et al., 2007). As only the curve index g is optimized in Π_2 , it is

unnecessary to compute the accurate curve evaluation $J_O(g)$. If we settle for obtaining a good enough curve from $\{\Gamma_g: g \in \mathbb{I}\}$, Π_2 can be solved efficiently by comparing the rough curve evaluations (biased but computationally easy) instead of $J_O(g)$.

The two-step framework of OODE is shown in Fig. 2. In step 1, for each candidate curve Γ_g ($g \in \mathbb{I}$), $J_1(g)$ is directly computed with \mathbf{p}_g , and then Ω_g is roughly solved by the DE algorithm. ‘Roughly’ means that a lower trajectory model resolution is applied: (1) The trajectory curve is divided more crudely. The number of trajectory segments takes N_c ($N_c < N$) instead of N , which reduces the problem size of Ω_g . (2) The vehicle body representation is simplified. The vehicle body is represented by K_c ($K_c < K$) circles, which simplifies the calculation of trajectory evaluation. Let $\mathbf{a}_{\text{copt},g}$ denote the rough solution of Ω_g after DE performs I iterations. The rough evaluation of Γ_g , denoted by $\hat{J}_O(g, I)$, is computed as

$$\hat{J}_O(g, I) = J_1(g) + J_2(g, \mathbf{a}_{\text{copt},g}). \tag{7}$$

Then the ‘best’ curve in $\{\Gamma_g: g \in \mathbb{I}\}$, denoted by $\Gamma_{\hat{g}}$, is determined by solving $\hat{g} = \arg \min_{g \in \mathbb{I}} \hat{J}_O(g, I)$. In step 2, the acceleration profile is optimized on curve $\Gamma_{\hat{g}}$. Then $\Omega_{\hat{g}}$ is ‘accurately’ solved by applying the original trajectory model resolution. Let $\mathbf{a}_{\hat{g}}$ denote the accurate solution of $\Omega_{\hat{g}}$. The solution finally returned by OODE is $(\mathbf{p}_{\hat{g}}, \mathbf{a}_{\hat{g}})$. OODE can obtain a ‘good enough’ trajectory curve without consuming too much time, and then focuses on optimizing the acceleration profile with the obtained curve.

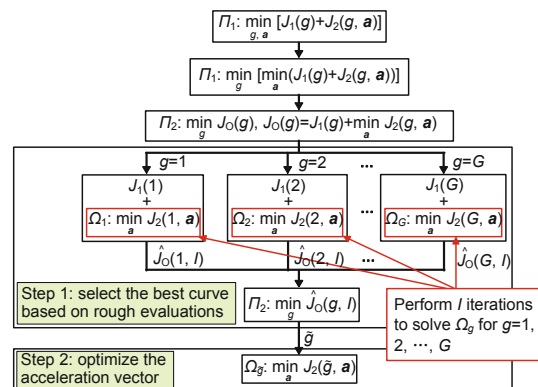


Fig. 2 Framework of ordinal-optimization-based differential evolution (OODE)

Note that OODE performs an equal number (I) of iterations in solving Ω_g for different g . In other words, during the computation of evaluation $\hat{J}_O(g, I)$, equal quantities of the computing budget are allocated to different curves. However, as $\hat{J}_O(g, I)$ becomes more accurate with the increase of I while we pick only the best curve, we can allocate more budget to the curves that are more likely to be wanted, so as to accelerate the identification of the best. Then the efficiency of OODE is enhanced.

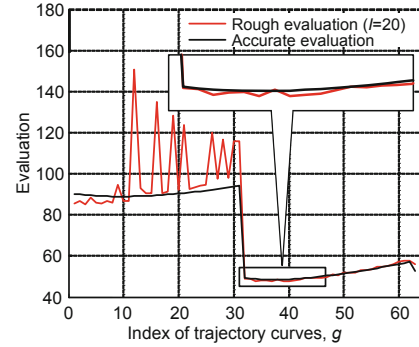
3 IOODE

3.1 Framework of IOODE

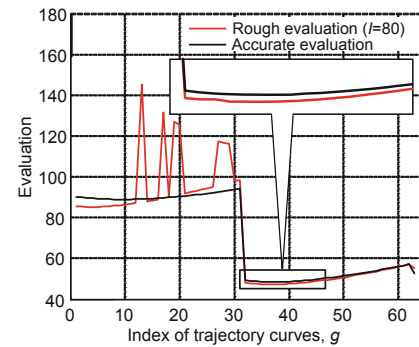
The algorithm IOODE is developed by introducing intelligent computing budget allocation into OODE. To show the feasibility of this idea, we first demonstrate the influence of rough curve evaluation on the solution quality. For a set of candidate curves $\{G_g: g = 1, 2, \dots, G\}$ within a given traffic scenario, we compute their accurate evaluations $J_O(g)$ by accurately solving Ω_g , so that the real rank of each curve in all curves is obtained. Then for each of these curves, four rough evaluations $\hat{J}_O(g, I)$ ($I = 20, 40, 60, 80$) are computed by roughly solving Ω_g . By comparing $\hat{J}_O(1, I), \hat{J}_O(2, I), \dots, \hat{J}_O(G, I)$, we pick the curve with the best rough evaluation and record its real rank. We perform this evaluating and comparing process 500 times, and use the top rank percentage (TRP) to denote the percentage of the experiments where the selected curve truly ranks in the top- $\xi\%$ of all curves. Then we can compute the TRPs of the above four rough evaluations at $\xi\% = 2\%, 4\%, \dots, 10\%$, which are listed in Table 2. An example of rough evaluations $\hat{J}_O(g, I)$ at $I = 20, 80, \infty$ and accurate evaluations $J_O(g)$ are given in Fig. 3, where curves 1–31 are right lane change curves, curves 32–62 are left lane change curves, and curve 63 is a lane keeping curve.

Table 2 Top rank percentages of four rough curve evaluations

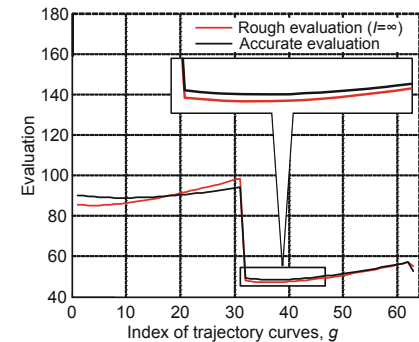
I	Top rank percentage				
	top-2%	top-4%	top-6%	top-8%	top-10%
20	23%	42%	66%	92%	96%
40	34%	80%	93%	100%	–
60	45%	99%	100%	–	–
80	17%	100%	–	–	–



(a)



(b)



(c)

Fig. 3 Evaluations of G ($G = 63$) candidate trajectory curves: (a) $\hat{J}_O(g, 20)$ vs. $J_O(g)$; (b) $\hat{J}_O(g, 80)$ vs. $J_O(g)$; (c) $\hat{J}_O(g, \infty)$ vs. $J_O(g)$

The relationship between rough and accurate curve evaluations can be described as

$$\hat{J}_O(g, I) = J_O(g) + E_c + E_i(I), \quad (8)$$

where E_c and $E_i(I)$ are the errors due to the reduced trajectory model resolution and insufficient iteration, respectively. Note that E_c depends on how much the model resolution is decreased, and $E_i(I)$ ($E_i(I) > 0$) tends to 0 as I goes to infinity. Table 2 and Fig. 3 show that the real rank of the selected curve determined by solving $\min_g \hat{J}_O(g, I)$ is improved in the

sense of probability with the increase of I . When $I = 80$, the selected curve truly ranks in the top-4% with probability 1.00. Note that the rough evaluations in Fig. 3a are enough to identify some curves which are impossible to be the truly best (e.g., curves 1–31), and thus we do not have to iterate 80 times for every curve.

We want to allocate the iterations intelligently so that the total number of iterations can be reduced. As shown in Fig. 4, we do the allocation in an iterative way. Let \mathbb{Q} represent the set of all candidate curves. A subset of \mathbb{Q} , denoted by \mathbb{P}_l , is defined to contain the promising curves which have potential to be selected as the best, with l being the number of performed allocation procedures. \mathbb{P}_l is initialized as $\mathbb{P}_0 = \mathbb{Q}$. Each time instead of \mathbb{Q} , we allocate only the computing budget to the elements in \mathbb{P}_l with I_c iterations performed for each curve. Thus, the rough evaluation $\hat{J}_O(g, \tau_g)$ is updated to be $\hat{J}_O(g, \tau_g + I_c)$, where τ_g is the number of iterations performed for curve Γ_g . After that, by comparing the evaluations of all candidate curves, the promising ones are picked out from \mathbb{Q} to make up the set \mathbb{P}_{l+1} . This process is repeated until only one promising curve is selected (i.e., $|\mathbb{P}_{l+1}| = 1$) or the largest allowed number of allocation procedures is reached (i.e., $l + 1 = L_{\max}$). Then the curve with the best rough evaluation is selected and its curve index is saved in \tilde{g} . Step 2 of IOODE is the same as the one of OODE.

Compared with OODE, IOODE always concentrates the computing resource on promising designs, and thus achieves higher efficiency. However, each time the promising curves should be carefully se-

lected to minimize the number of selected curves. We should also ensure that the truly best curve is selected with a high enough probability.

As I increases, $\hat{J}_O(g, I)$ converges to the true evaluation $\hat{J}_O(g, \infty)$. For the purpose of comparing curves which have executed various numbers of iterations, it is natural to predict $\hat{J}_O(g, \infty)$ based on the values of $\hat{J}_O(g, I)$ at $I = 1, 2, \dots, \tau_g$ for each curve Γ_g . Then based on the prediction results we determine the promising curves in \mathbb{Q} . Thus, the evaluation prediction model (EPM) is developed. The prediction result is denoted by Y_g . Then because of the random prediction error in Y_g , EPM further computes the posterior probability distribution (PD) of $\hat{J}_O(g, \infty)$, which is denoted by the PD of random variable X_g . Each time, Y_g and X_g of the curves in \mathbb{P}_l are updated and sent into the curve selection model (CSM). CSM determines \mathbb{P}_{l+1} based on the probability that the truly best is contained in \mathbb{P}_{l+1} . This process is shown in Fig. 5. Details of EPM and CSM will be given in the next two subsections.

Figs. 4 and 5 show how to iteratively allocate the computing budget to promising designs, which can also be generalized to other candidate-curve-based planning methods. The key is the EPM whose evaluation accuracy depends on the consumed computational resource, and the CSM which determines the promising designs based on uncertain or biased evaluations.

3.2 Predicting rough curve evaluation

The promising curves in \mathbb{Q} are determined by comparing $\hat{J}_O(g, \infty)$. Because $\hat{J}_O(g, I)$ converges when I increases, we can predict the limit of the sequence $\{z(n)\}$ ($z(n) = \hat{J}_O(g, n)$) as the estimation of $\hat{J}_O(g, \infty)$.

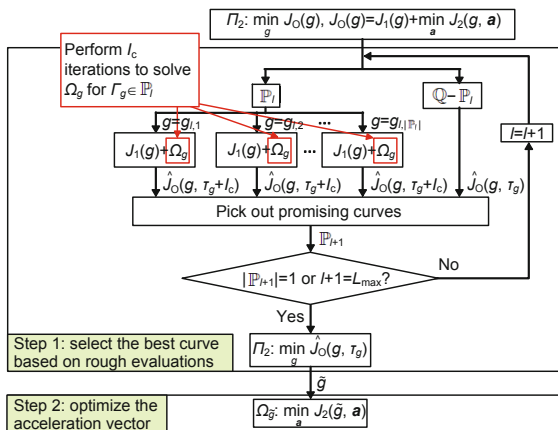


Fig. 4 Framework of IOODE

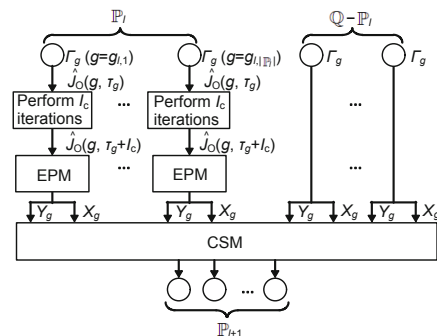


Fig. 5 Picking out the promising curves

For curve Γ_g in \mathbb{P}_l , since τ_g iterations have been performed in roughly solving Ω_g , we can compute the rough curve evaluation $\hat{J}_O(g, I)$ at $I = 1, 2, \dots, \tau_g$ with Eq. (7), so the sequence $\{z(n) : n = 1, 2, \dots, N, N = \tau_g\}$ is obtained. EPM uses a power function curve $z = \hat{z}(\theta, n)$ to fit the points $(1, z(1)), (2, z(2)), \dots, (N, z(N))$ in the Cartesian n - z coordinate plane, and computes $\hat{z}(\theta, \infty)$ as the estimate of $z(\infty)$. Here, $\hat{z}(\theta, n)$ is defined as

$$\hat{z}(\theta, n) = A \cdot n^{-B} + C, \quad (9)$$

where $\theta = [A \ B \ C]$, with A, B , and C all being positive real numbers. The relationship between $\hat{z}(\theta, \infty)$ and $z(\infty)$ is shown in Fig. 6, where the green points represent $(1, z(1)), (2, z(2)), \dots, (N, z(N))$, and the red points represent $(N+1, z(N+1)), (N+2, z(N+2)), \dots, (\infty, z(\infty))$.

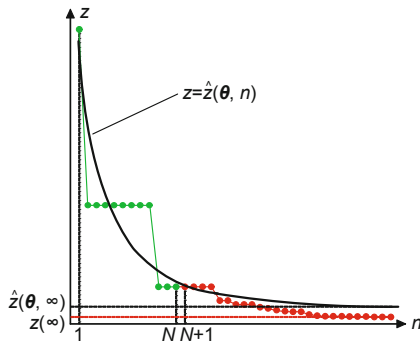


Fig. 6 Prediction of curve evaluation. References to color refer to the online version of this figure

The fitting error is defined to be $W_N(\theta) = \sum_{n=1}^N w_n \cdot (z(n) - \hat{z}(\theta, n))^2$, where w_n is the weight of the n th error. Thus, the fitting goal is to compute $\theta = \underset{\theta}{\operatorname{argmin}} W_N(\theta)$. To maximize computing efficiency, EPM is designed to work in an iterative way. With n starting from one, every time a new point $(n, z(n))$ comes, θ_{n-1} is updated as $\theta_n = \theta_{n-1} + \Delta\theta_n$. The derivation of $\theta_n = \theta_{n-1} + \Delta\theta_n$ can be found in Bai *et al.* (2012). Then for curve Γ_g , $\hat{z}(\theta, \infty)|_{\theta_N}$ is the prediction result for $\hat{J}_O(g, \infty)$, which is denoted by Y_g .

However, there exists prediction error ERR between Y_g and $\hat{J}_O(g, \infty)$. We have

$$Y_g = \hat{J}_O(g, \infty) + \text{ERR}, \quad (10)$$

where ERR is stochastic due to the randomness of the elements in $\{z(n)\}$. In this study, we use the em-

pirical distribution of ERR, which is obtained offline, to compute the posterior PD of $\hat{J}_O(g, \infty)$. Totally 1000 trajectory curves that start and end with random pose vectors are generated. For each of these curves (Γ_g), $\hat{J}_O(g, \infty)$ is computed. Then EPM is applied to predict Y_g based on different numbers of iterations (I) and compute $\text{ERR} = Y_g - \hat{J}_O(g, \infty)$. Consequently, with a given I , we have 1000 samples of ERR, so that the empirical distribution of ERR can be computed. Fig. 7 shows the empirical distributions of ERR at $I = 20, 40, 60, 80$. The variance of ERR decreases with the increase of I . By investigating the samples of ERR, we also find that the randomness of ERR is mainly due to the fact that DE is a random algorithm, but is not very related to the curve being evaluated. Thus, the discrete distributions in Fig. 7 are directly used to compute the posterior PD of $\hat{J}_O(g, \infty)$ (denoted by the PD of X_g , $X_g = Y_g - \text{ERR}$).

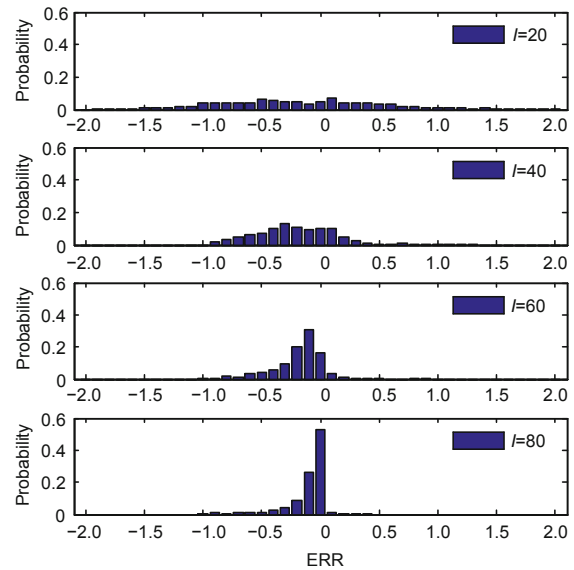


Fig. 7 Empirical distributions of ERR

3.3 Selecting the promising curves

For each curve Γ_g , both the predicted rough evaluation (Y_g) and the posterior PD of the true rough evaluation (PD of X_g) are known. CSM is applied to select the promising curves from \mathbb{Q} .

Let $Y_{[1]} \leq Y_{[2]} \leq \dots \leq Y_{[G]}$ be the ordered predicted evaluations of candidate curves. Let $X_{[1]}, X_{[2]}, \dots, X_{[G]}$ be a permutation of the variables X_1, X_2, \dots, X_G , where $X_{[g]}$ corresponds to the

same curve as $Y_{[g]}$ for $g=1, 2, \dots, G$. Note that $X_{[1]}, X_{[2]}, \dots, X_{[G]}$ may not satisfy $X_{[1]} \leq X_{[2]} \leq \dots \leq X_{[G]}$ due to the error ERR. Then the curves corresponding to $Y_{[1]}, Y_{[2]}, \dots, Y_{[q]}$ ($1 \leq q \leq G$) are selected as the promising curves. A measurement of selection quality is defined to be the probability that the truly best (the smallest element) of $X_{[1]}, X_{[2]}, \dots, X_{[G]}$, denoted by X_{best} , is included in the set $\{X_{[1]}, X_{[2]}, \dots, X_{[q]}\}$. Therefore, the goal is to minimize q with a high enough value of this probability ensured, i.e.,

$$\begin{aligned} A: \quad & \min q, \quad q \in \mathbb{N}, \quad 1 \leq q \leq P \\ \text{s.t.} \quad & P\{X_{\text{best}} \in \{X_{[1]}, X_{[2]}, \dots, X_{[q]}\}\} \geq r\%, \end{aligned} \quad (11)$$

where $P\{E\}$ is the probability that the event E occurs.

We define

$$\begin{aligned} P_B &\triangleq P\{X_{\text{best}} \in \{X_{[1]}, X_{[2]}, \dots, X_{[q]}\}\} \\ &= 1 - P\{X_{\text{best}} \in \{X_{[q+1]}, X_{[q+2]}, \dots, X_{[G]}\}\} \\ &= 1 - \sum_{g=q+1}^G P\{X_{\text{best}} = X_{[g]}\}. \end{aligned} \quad (12)$$

With the discrete distributions of $X_{[1]}, X_{[2]}, \dots, X_{[G]}$, we have

$$\begin{aligned} P\{X_{\text{best}} = X_{[g]}\} &= \sum_r \left[P\{m_r \leq X_{[g]} < m_{r+1}\} \right. \\ &\quad \cdot \left. \prod_{p=1, p \neq g}^G P\{X_{[p]} \geq m_{r+1}\} \right], \end{aligned} \quad (13)$$

where $[m_r, m_{r+1}]$ is the r th interval of $X_{[g]}$ in its distribution. Thus, with a given q , P_B can be computed with Eqs. (12) and (13).

For the integer programming problem A , the computing burden of P_B is not heavy. Thus, a ‘trial and error’ method is applied to solve it. The C style pseudo code is shown in Algorithm 1.

Algorithm 1 Algorithm for solving A

```

1:  $P_B \leftarrow 1$  // Initialize  $P_B$ 
2:  $g \leftarrow G$  // Initialize the solution
3: loop
4:    $P_B \leftarrow P_B - P\{X_{\text{best}} = X_{[g]}\}$ 
5:   if  $P_B < r\%$  then
6:     break
7:   end if
8:    $g \leftarrow g - 1$ 
9: end loop
10:  $q \leftarrow g$  // Return the solution

```

The ‘trial and error’ method cannot guarantee that the obtained q is globally optimal, but the solution is sufficient for the fast selection of promising curves from candidates.

4 Simulation results

In this section, the planning results and performance of IOODE are demonstrated and analyzed. All the simulations are performed in Visual Studio 2010 using an Intel® Core™ i5 CPU M 480@2.67 GHz with 2.0 GB RAM. A typical traffic scenario on straight roads, which includes three dynamic obstacles (vehicles) α, β, γ , is considered. All obstacles are assumed to move along the lane centerlines at constant speeds. The initial state of the scenario is denoted by a set of scenario parameters, whose meanings are shown in Fig. 8. Parameters $\kappa_0, \theta_0, y_0, v_0$, and a_0 are the Ego vehicle (EV)’s initial curvature, orientation, lateral position, velocity, and acceleration, respectively. Parameters $x_\alpha, x_\beta, x_\gamma$ are the longitudinal distances between α, β, γ and EV, respectively. Parameters v_α, v_β , and v_γ are the velocities of α, β , and γ , respectively. A problem instance of trajectory planning can be represented by the scenario parameters. We simulate four specific scenarios (S1, S2, S3, S4), and introduce a test set of 2000 scenarios, whose scenario parameters take random values. Let SR denote such a random scenario. The parameters of S1, S2, S3, S4, and SR are given in Table 3. The parameters of IOODE are listed in Table 4.

4.1 Trajectory planning results

IOODE is applied to plan trajectories in scenarios S1 and S2. The planning results are shown in Figs. 9 and 10, respectively. Figs. 9a and 10a demonstrate the positions of obstacles and EV at

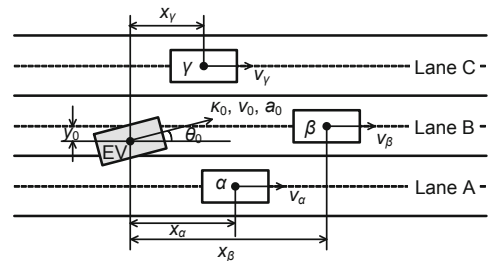


Fig. 8 Initial state of the traffic scenario

$t = 0, 0.5, 1.0, 1.5$ s (from top to bottom). The blue line is the generated trajectory curve with a red circle and a blue star representing the start and end points, respectively. The bodies of EV and obstacles are denoted by gray and white rectangles, respectively. Figs. 9b and 10b demonstrate the generated acceleration profile where the 25 blue blocks represent the accelerations in 25 trajectory segments.

In scenario S1, the preceding vehicle β moves

Table 3 Parameters of traffic scenarios

Parameter	Parameter value				
	S1	S2	S3	S4	SR
y_0 (feet)	-2	3	5	-3	$\mathcal{U}(-6, 6)$
θ_0 (rad)	0.1	-0.1	0.3	-0.4	$\mathcal{U}(-0.5, 0.5)$
κ_0 (1/feet)	0	0	0.01	0.02	$\mathcal{U}(-0.02, 0.02)$
v_0 (feet/s)	35	30	40	20	$\mathcal{U}(10, 60)$
a_0 (feet/s ²)	0	0	4	-2	$\mathcal{U}(-6, 6)$
x_α (feet)	10	0	-50	10	$\mathcal{U}(-100, 100)$
v_α (feet/s)	30	50	30	50	$\mathcal{U}(10, 60)$
x_β (feet)	50	50	80	160	$\mathcal{U}(50, 200)$
v_β (feet/s)	20	30	40	50	$\mathcal{U}(10, 60)$
x_γ (feet)	-30	20	-80	-20	$\mathcal{U}(-100, 100)$
v_γ (feet/s)	10	30	50	20	$\mathcal{U}(10, 60)$

$\mathcal{U}(a, b)$ represents the uniform distribution on interval $[a, b]$

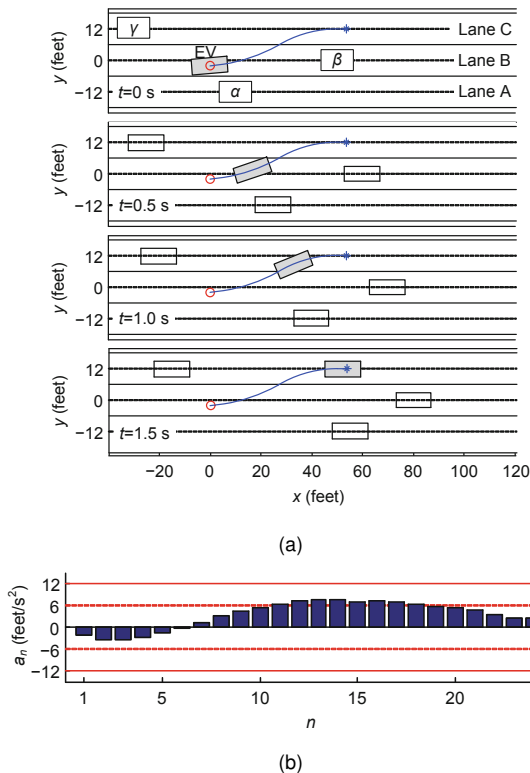


Fig. 9 Planning results in scenario S1: (a) trajectory curve; (b) acceleration profile. References to color refer to the online version of this figure

slower than EV. Since Lane A is occupied by α which stops EV from entering but Lane C is not, the optimal trajectory for EV is to perform a lane change into the left adjacent lane. EV first has to decelerate to keep a safe distance from β , but afterwards EV can accelerate (Fig. 9b) to enter Lane C. While in scenario S2, since Lane A and Lane C are both occupied, EV can stay only in Lane B and adjust its speed to follow β (Fig. 10b).

In addition, we simulate IOODE in a rolling horizon framework. The replanning cycle T_0 can be as small as possible but should be larger than the time consumed for computing a trajectory. Here, we set $T_0 = 0.3$ s. Fig. 11 gives an example, where EV overtakes a slower moving vehicle in the middle lane by performing a double lane change maneuver.

4.2 Computing budget allocation analysis

The processes of computing budget allocation for trajectory planning in scenarios S1 and S2 are depicted in Figs. 12 and 13, respectively. For scenario S1, the best curve is determined after four

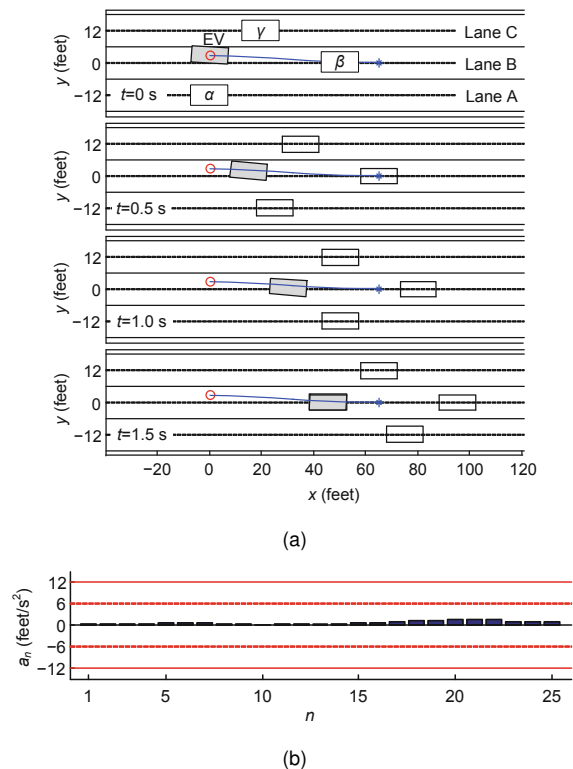


Fig. 10 Planning results in scenario S2: (a) trajectory curve; (b) acceleration profile. References to color refer to the online version of this figure

rounds of iterative budget allocation. The posterior PDs of the rough curve evaluation $\hat{J}_O(g, \infty)$ at $l = 0, 1, 2, 3$ are shown in Fig. 12. Each short line depicts the interval where the evaluation possibly takes its value. The red lines marked with 'x' represent the selected promising curves (i.e., \mathbb{P}_{l+1}), while the blue lines represent the remaining ones (i.e., $\mathbb{Q} - \mathbb{P}_{l+1}$). For example, when $l = 0$, $14/63 \approx 22.2\%$ of the curves are selected and the computing budget will be allocated to them. When $l = 3$, the curve with the best rough evaluation is selected as the best curve.

For scenario S2, the best curve is determined with only one round of budget allocation performed. The symbols in Fig. 13 are identically defined as in Fig. 12. Curve 63 is significantly better than the rest, and thus it is directly selected even though only 20 iterations are performed.

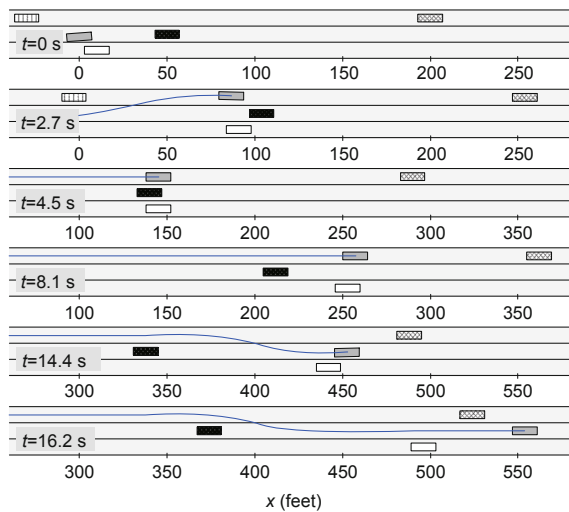


Fig. 11 IOODE performed in a rolling horizon framework with a replanning cycle $T_0 = 0.3$ s

4.3 Algorithm performance analysis

We apply OODE and IOODE to solve the 2000 random problem instances in the test set. OODE applies the framework in Fig. 2 to solve Π_1 , where each candidate curve is roughly evaluated by executing 80 iterations (equal to the largest possible number of iterations for evaluating a curve in IOODE). The real rank of the 'best' curve picked by the two algorithms is investigated. We can compute the percentage of the results where the selected curve truly ranks in the top- $\xi\%$, i.e., TRP. Fig. 14 shows the relationship between $\xi\%$ and TRP. The curves of OODE and IOODE are very close, which means no loss of solution quality is induced with the introduction of intelligent computing budget allocation. IOODE ob-

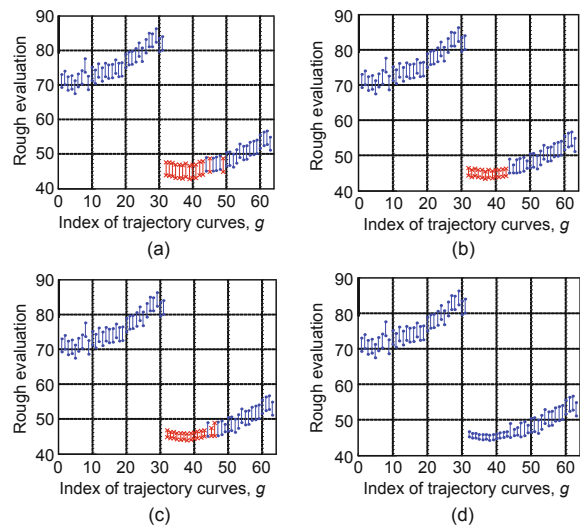


Fig. 12 Computing budget allocation for trajectory planning in scenario S1: (a) $l = 0$; (b) $l = 1$; (c) $l = 2$; (d) $l = 3$. References to color refer to the online version of this figure

Table 4 Parameters of IOODE

Parameter	Value	Interpretation
G	63	Total number of candidate trajectory curves
K	5	Number of circles used to accurately represent EV's body
K_c	1	Number of circles used to roughly represent EV's body
N	25	Number of segments, into which the trajectory is accurately divided
N_c	5	Number of segments, into which the trajectory is roughly divided
NP_c	10	Population size of DE
F_c	0.85	Differential weight of DE
CR_c	0.95	Crossover probability of DE
I_c	20	Number of iterations allocated to each selected promising curve
L_{\max}	4	Largest allowed number of budget allocation procedures
$r\%$	99.9%	Lowest accepted probability that the truly best curve is included in the selected curves

tains a top-5% trajectory curve with probability 0.95, and a top-10% curve with probability 0.97.

Then we compare the performance of three algorithms (OODE, IOODE, and the traditional intelligent optimization algorithm DE), which are applied to solve the problem instances in scenarios S1–S4 and the test set. The results are given in Table 5. DE assumes that the curve parameter \mathbf{p} takes its value in the minimum continuous set that contains $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_G$, and then directly optimizes \mathbf{p} and \mathbf{a} .

DE optimizes the trajectory curve continuously while OODE and IOODE discretize the continuous space into a limited number of candidate curves. Thus, DE statistically returns a better solution than OODE and IOODE (see the column ‘ \bar{J} for SR’). However, due to the complexity of the high-dimensional, non-convex solution space, DE consumes much more time and sometimes is trapped in a local optimum (e.g., ‘ J for S4’). We see no significant difference between the solution quality of IOODE and OODE, but the average planning speed of IOODE is about 20% faster than that of OODE. In the worst case (i.e., all candidate curves have equal evaluation values), IOODE would deteriorate to be the equal allocation schema, the same as OODE. However, this is rare in real applications.

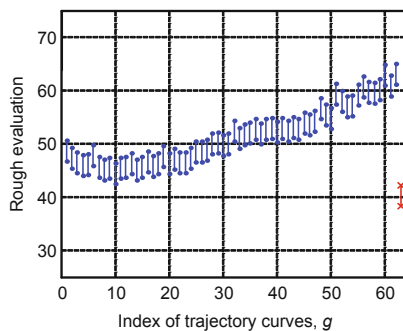


Fig. 13 Computing budget allocation for trajectory planning in scenario S2. References to color refer to the online version of this figure

IOODE generates a trajectory of time length 1.5 s in 200 ms. During the planning period, the largest position estimation error for an obstacle due to the constant speed assumption is $0.5 \times 3.5 \text{ m/s}^2 \times (0.2 \text{ s})^2 = 0.07 \text{ m}$, where 3.5 m/s^2 is the obstacle’s largest acceleration/deceleration. Consequently, this assumption is believed to be valid and IOODE is capable of working in real time.

5 Conclusions

We propose a novel candidate-curve-based trajectory planning algorithm named ‘IOODE’ by introducing intelligent computing budget allocation into OODE. When selecting the best curve based on curve evaluation, IOODE iteratively allocates the computing budget to promising designs, which effectively reduces the computing cost. The simulation results show that IOODE is about 20% faster than OODE with no loss of solution quality. In addition, the framework of computing budget allocation shown here can be applied to other candidate-curve-based planning methods. The key is to build an evaluation model whose accuracy depends on the computing cost and a selection model which selects good designs based on biased evaluations.

However, due to the fact that IOODE spends at

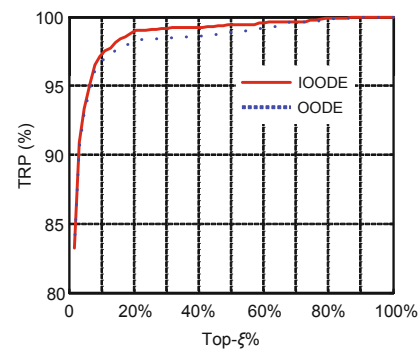


Fig. 14 Top rank percentage (TRP) vs. $\xi\%$ using IOODE and OODE to select the best curve

Table 5 Algorithm performances of DE, OODE, and IOODE

Algorithm	J for S1	J for S2	J for S3	J for S4	\bar{J} for SR	\bar{T} for SR (ms)
DE	43.66	53.00	30.15	35.32	38.19	2449.9
OODE	46.98	52.86	30.30	32.15	40.55	244.3
IOODE	47.02	52.84	30.35	32.19	40.58	198.8

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 represents the mean evaluation of the solutions of the problems in the test set; \bar{T} for SR represents the mean CPU time for solving the problems in the test set

least 80% of the CPU time on optimizing the acceleration vector (step 2), it is no longer easy to further improve the planning speed by optimizing curve selection (step 1). Fortunately, though with a lower model resolution, the problem $\Omega_{\tilde{g}}$ which needs to be solved in step 2, has been roughly solved once in step 1. Our future work will focus on applying more knowledge from the optimization result of step 1 to assist the optimization of the acceleration.

References

- Bai, L., Jiang, Y., Huang, D., 2012. A novel two-level optimization framework based on constrained ordinal optimization and evolutionary algorithms for scheduling of multipipeline crude oil blending. *Ind. Eng. Chem. Res.*, **51**(26):9078-9093.
<http://dx.doi.org/10.1021/ie202224w>
- Bechhofer, R.E., Santner, T.J., Goldsman, D.M., 1995. Design and Analysis of Experiments for Statistical Selection, Screening, and Multiple Comparisons. Wiley, New York, USA.
- Bengler, K., Dietmayer, K., Farber, B., et al., 2014. Three decades of driver assistance systems: review and future perspectives. *IEEE Intell. Transp. Syst. Mag.*, **6**(4):6-22. <http://dx.doi.org/10.1109/MITS.2014.2336271>
- Branke, J., Chick, S.E., Schmidt, C., 2007. Selecting a selection procedure. *Manag. Sci.*, **53**(12):1916-1932. <http://dx.doi.org/10.1287/mnsc.1070.0721>
- Chen, C., Lee, L.H., 2010. Stochastic Simulation Optimization: an Optimal Computing Budget Allocation. World Scientific, USA.
- Chen, C., Yüesan, E., 2005. An alternative simulation budget allocation scheme for efficient simulation. *Int. J. Simul. Process Model.*, **1**(1/2):49-57. <http://dx.doi.org/10.1504/IJSPM.2005.007113>
- Chen, C., Lin, J., Yücesan, E., et al., 2000. Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Discr. Event Dyn. Syst.*, **10**(3):251-270. <http://dx.doi.org/10.1023/A:1008349927281>
- Chen, C., Chick, S.E., Lee, L.H., et al., 2015. Ranking and selection: efficient simulation budget allocation. In: Fu, M.C. (Ed.), Handbook of Simulation Optimization. Springer, New York, USA. http://dx.doi.org/10.1007/978-1-4939-1384-8_3
- Chick, S.E., Inoue, K., 2001. New two-stage and sequential procedures for selecting the best simulated system. *Oper. Res.*, **49**(5):732-743. <http://dx.doi.org/10.1287/opre.49.5.732.10615>
- Chu, K., Lee, M., Sunwoo, M., 2012. Local path planning for off-road autonomous driving with avoidance of static obstacles. *IEEE Trans. Intell. Transp. Syst.*, **13**(4):1599-1616. <http://dx.doi.org/10.1109/TITS.2012.2198214>
- Fu, X., Jiang, Y., Huang, D., et al., 2015. A novel real-time trajectory planning algorithm for intelligent vehicles. *Contr. Dec.*, **30**(10):1751-1758 (in Chinese).
- Gehrig, S.K., Stein, F.J., 2007. Collision avoidance for vehicle-following systems. *IEEE Trans. Intell. Transp. Syst.*, **8**(2):233-244. <http://dx.doi.org/10.1109/TITS.2006.888594>
- Glaser, S., Vanholme, B., Mammarr, S., et al., 2010. Maneuver-based trajectory planning for highly autonomous vehicles on real road with traffic and driver interaction. *IEEE Trans. Intell. Transp. Syst.*, **11**(3):589-606. <http://dx.doi.org/10.1109/TITS.2010.2046037>
- Hilgert, J., Hirsch, K., Bertram, T., et al., 2003. Emergency path planning for autonomous vehicles using elastic band theory. Proc. IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics, p.1390-1395. <http://dx.doi.org/10.1109/AIM.2003.1225546>
- Ho, Y., Zhao, Q., Jia, Q., 2007. Ordinal Optimization: Soft Optimization for Hard Problems. Springer, New York, USA. <http://dx.doi.org/10.1007/978-0-387-68692-9>
- Kim, S., Nelson, B.L., 2001. A fully sequential procedure for indifference-zone selection in simulation. *ACM Trans. Model. Comput. Simul.*, **11**(3):251-273. <http://dx.doi.org/10.1145/502109.502111>
- Köhler, S., Schreiner, B., Ronalter, S., et al., 2013. Autonomous evasive maneuvers triggered by infrastructure-based detection of pedestrian intentions. Proc. IEEE Intelligent Vehicles Symp., p.519-526. <http://dx.doi.org/10.1109/IVS.2013.6629520>
- Kuwata, Y., Teo, J., Fiore, G., et al., 2009. Real-time motion planning with applications to autonomous urban driving. *IEEE Trans. Contr. Syst. Technol.*, **17**(5):1105-1118. <http://dx.doi.org/10.1109/TCST.2008.2012116>
- Ma, L., Xue, J., Kawabata, K., et al., 2015. Efficient sampling-based motion planning for on-road autonomous driving. *IEEE Trans. Intell. Transp. Syst.*, **16**(4):1961-1976. <http://dx.doi.org/10.1109/TITS.2015.2389215>
- McNaughton, M., Urmson, C., Dolan, J.M., et al., 2011. Motion planning for autonomous driving with a conformal spatiotemporal lattice. Proc. IEEE Int. Conf. on Robotics and Automation, p.4889-4895. <http://dx.doi.org/10.1109/ICRA.2011.5980223>
- Montemerlo, M., Becker, J., Bhat, S., et al., 2008. Junior: the Stanford entry in the urban challenge. *J. Field Robot.*, **25**(9):569-597. <http://dx.doi.org/10.1002/rob.20258>
- Papadimitriou, I., Tomizuka, M., 2003. Fast lane changing computations using polynomials. Proc. American Control Conf., p.48-53. <http://dx.doi.org/10.1109/ACC.2003.1238912>
- Reif, J.H., 1979. Complexity of the mover's problem and generalizations. Proc. 20th Annual Symp. on Foundations of Computer Science, p.421-427. <http://dx.doi.org/10.1109/SFCS.1979.10>
- Urmson, C., Anhalt, J., Bagnell, D., et al., 2008. Autonomous driving in urban environments: boss and the urban challenge. *J. Field Robot.*, **25**(8):425-466. <http://dx.doi.org/10.1002/rob.20255>
- Ziegler, J., Stiller, C., 2009. Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios. Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, p.1879-1884. <http://dx.doi.org/10.1109/IROS.2009.5354448>