Title: Simultaneous versus joint computing: a case study of multi-vehicle parking motion planning

Authors: Bai Li, Youmin Zhang, Zhijiang Shao, Ning Jia

PII: S1877-7503(16)30254-X
DOI: http://dx.doi.org/doi:10.1016/j.jocs.2017.03.015
Reference: JOCS 637

To appear in:

Received date: 23-10-2016
Revised date: 13-2-2017
Accepted date: 18-3-2017

Simultaneous versus joint computing: a case study of multi-vehicle parking motion planning

Bai Li [a *], Youmin Zhang [b], Zhijiang Shao [a c], Ning Jia [d]

a. College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China
b. Department of Mechanical and Industrial Engineering, Concordia University, Montreal, Quebec H3G 1M8, Canada
c. State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China
d. Institute of Systems Engineering, Tianjin University, Tianjin 300072, China

* Corresponding author, Tel. +86 571 87953068
libaioutstanding@163.com, libai@zju.edu.cn (B. Li); ymzhang@encs.concordia.ca (Y. Zhang); szj@zju.edu.cn (Z. Shao); jia_ning@tju.edu.cn (N. Jia).

Highlights

- Prevalent simultaneous and joint computation methods for multi-vehicle motion planning problems are comprehensively reviewed.
- Multi-vehicle motion planning problem is described as a generic optimal control problem.
- An initialization methodology is proposed to facilitate the process in numerically solving optimal control problem.
- Comparisons are made to show the advantages of simultaneous computation over joint computation.

Abstract

Multi-vehicle motion planning (MVMP) refers to computing feasible trajectories for multiple vehicles. MVMP problems are generally solved in two ways, namely *simultaneous methods* and *joint methods*. An inherent difference between both types of methods is that, simultaneous methods compute motions for vehicles all at once, while joint methods divide the original problem into parts and combine them together. The joint methods usually sacrifice solution quality for computational efficiency, and the simultaneous methods are applicable to simple or simplified scenarios only. These defects motivate us to develop an efficient simultaneous computation method which provides high-quality solutions in generic cases. Progressively constrained dynamic optimization (PCDO), an initialization-based computation framework is proposed to ease the burdens of simultaneous computation methodologies when they are adopted to solve the MVMP problem problems. Specifically, PCDO locates and discards the redundant constraints in the MVMP problem formulation so as to reduce the problem scale, thereby easing the problem-solving process. Our simulations focus on the cooperative parking scheme of automated vehicles. Comparative simulation results show that 1) the designs in PCDO are efficient, and

2) simultaneous computation outperforms joint computation.

Keywords: multi-vehicle motion planning (MVMP); automated vehicle; simultaneous computation; nonlinear programming; progressively constrained dynamic optimization (PCDO)

1. Introduction

Single-vehicle motion planning refers to finding a feasible trajectory from a starting configuration towards a specified destination with various restrictions (e.g., collision avoidance, and kinematics/dynamics) considered [1]. A natural extension of this problem is multi-vehicle motion planning (MVMP), which is more than just planning motions for individual vehicles jointly because of the large-scale interactions among the vehicles. With the backdrop of vehicle automation and connection, applications of MVMP have been widely developed [2], e.g., in cooperative terrain exploration [3], cooperative bike reposition [4], and cooperative lane change [5,6].

In solving MVMP problems, nominally a centralized system should be established, which contains the states of all the involved vehicles. Through computation on the basis of such a centralized system, the motion of each vehicle is obtained. This centralized formulation, nonetheless, is computationally expensive: suppose there are $\mathrm{N_v}$ vehicles, then there would be $\mathrm{C}_{\mathrm{N_v}}^{2}$ interaction related constraints (e.g., collision avoidance) imposed at *every* moment when vehicles move. Since $\mathrm{C}_{\mathrm{N_v}}^{2}$ exponentially grows with $\mathrm{N_v}$, solving one MVMP problem with $\mathrm{N_v}$ vehicles is more challenging than handling $\mathrm{N_v}$ individual vehicles altogether. Remedies for this limitation have been considered in two ways: one is to decentralize the original MVMP problem into new problems, and the other is to develop powerful solvers that can directly tackle the original problem.

Decentralized methods usually run fast because they no longer consider the full configuration space as the centralized MVMP methods do. Typical branches of decentralized methods include *prioritization-*, *coordination-*, and *partition-based* approaches. Prioritization approaches plan motion for each vehicle sequentially in a specified order. Buckley [7] indicated that careful priority assignments help to improve planning efficiency. Thus in his work, the priority is assigned such that the number of vehicles that can move in straight lines is maximized. Liu et al. [8] developed a dynamic priority assignment principle: if the (estimated) trajectory of vehicle 1 is blocked by the initial/terminal location of vehicle 2, then vehicle 1 yields its priority to vehicle 2. Plessen et al. [9] proposed a priority swapping strategy so as to maximize the traffic throughput. Coordination, as the second branch, is about attaching time courses to the given paths so as to form feasible motions. Herein, "feasible" stands for collision-free in most of the related references [10,11]. Typically, sample-based path planners are adopted to generate the "given paths" because they run fast (e.g., [1,12,13]). The third (partition-based) branch is about dividing the centralized MVMP formulation into multiple parts, stages, or levels, e.g., task assignment + motion planning [14], task assignment + motion coordination [15], and subgrouping + motion planning [16,17]. The priorities in the first branch, if not appropriately assigned, weaken the cooperative capability of a vehicle team. In some complicated scenarios, estimating priorities is not easier than solving the original centralized MVMP problem. An inherent limitation of the second branch is that, it inevitably incurs failures (such as deadlock) in complicated scenarios. This is because the path generation stage considers nothing related to time-dependent constraints. Approaches in the third branch are only suitable for sparse-space problems, e.g., unmanned aerial vehicle (UAV) related tasks, wherein the task allocation estimation may be always near-optimal. However, the MVMP quality would be influenced if the allocation/subgrouping estimation is poor, when the scenarios are not sparse.

In addition to decentralization, the other way aims to facilitate the process in solving the original MVMP problem. Mathematical programming methods and metaheuristic methods have been developed. All the metaheuristic algorithms, as far as the authors know, can handle decision variables with bounded constraints only, but not with complicated constraints of any other type. Since the motion feasibility related constraints in MVMP problems are highly complicated, they have to be formulated as part of the optimization objective (i.e., the so-called penalty functions) instead of being constraints

[18,19]. Naturally, converting constraints to objectives would result in extremely long computational time before solution feasibility is achieved. Thus metaheuristic algorithms are not qualified to handle generic MVMP problems at all. Compared to metaheuristics, mathematical programming methods perform better in handling constraints [20]. Häusler et al. [21] applied projection operator approach for motion planning of multiple marine vehicles; Richards and How [22] applied CPLEX for multiple aircraft motion planning; Borrelli et al. [23] applied IPOPT for trajectory planning of multi-UAVs; Abichandani et al. [24] presented an interesting survey of the prevalent mathematical programming methodologies used for MVMP problems. However, the existing programming formulations are simple or simplified. In more details, [21] and [23] formulated fixed-time (rather than free-time) multi-vehicle motion planning problems, wherein the vehicle kinematics was overly simplified and particularly in [21], the underwater vehicles were simply assumed as circular; [22] adopted simplified problem formulations without nonlinearity. There have been none methods that handle the high nonlinearity and large scales in generic MVMP problems.

As a brief summary, decentralized motion planning methodologies pursue for computational efficiency at the sacrifice of solution quality. Centralized motion planners directly solve the original MVMP problem but are limited to simple or simplified scenarios. An inherent difference between centralized and decentralized methods is that, centralized methods compute motions for all the vehicles *simultaneously*, whereas decentralized methods do it *jointly*, that is, divide the full configuration space into separated sub-spaces, and then combine the solutions to each sub-space together as the overall solution. A collection of the aforementioned references, together with [25–28], is depicted in **Fig. 1**. Although cooperation has been mentioned in the titles of those references, quite few studies really cared about exploiting the cooperation potential in a multi-vehicle team. The decentralization efforts in the joint methods and the simplification efforts in the simultaneous methods largely limit the "cooperation talent". This motivates us to develop a computationally efficient strategy that can ease solving the original centralized MVMP problems, whereby the cooperation capability of multiple vehicles is maximized.

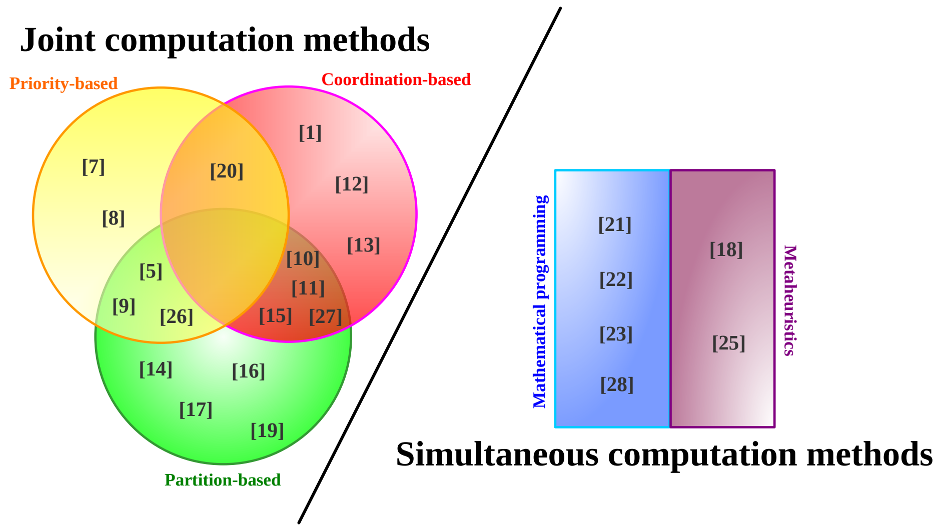


Fig. 1. A comprehensive review on references related to MVMP problems.

Highlights of this work lie in that 1) generic MVMP problems are formulated as centralized optimal control problems with collision-avoidance constraints clearly defined; 2) an initialization-based computation framework is proposed to ease the burdens of simultaneous computation methodologies when they are adopted to solve the optimal control problems.

The rest of this paper is organized as follows. Section 2 describes MVMP problems as generic nonlinear optimal control problems, and the collaborative parking is considered as the application. Section 3 introduces the initialization-based computation framework. Simulation results are reported in Section 4, together with in-depth discussions. Conclusions are drawn in Section 5.

2. MVMP problem formulation

In this section, generic MVMP problems are formulated as optimal control problems.

2.1 Vehicle kinematics

We consider the kinematics of each vehicle as follows [29]:

$$\begin{cases} \dot{x}_i(t) = v_i(t)\cdot\cos\theta_i(t) \\ \dot{y}_i(t) = v_i(t)\cdot\sin\theta_i(t) \\ \dot{v}_i(t) = a_i(t) \\ \dot{\theta}_i(t) = v_i(t)\cdot\tan\phi_i(t)/\mathrm{L_w} \\ \dot{\phi}_i(t) = \omega_i(t) \end{cases}, \ t\in[0,t_{\mathrm{f}}],\ i=1,2,\ldots,\mathrm{N_v}, \tag{1}$$

where t_{f} stands for the completion time, $(x_i, y_i) = P_i$ locates the rear wheel axis midpoint of the vehicle (**Fig. 2**), $\theta_i(t)$ denotes the orientation angle, $v_i(t)$ refers to the longitudinal velocity, $a_i(t)$ denotes the longitudinal acceleration, $\phi_i(t)$ denotes the steering angle, $\omega_i(t)$ denotes the corresponding angular velocity, and $\mathrm{N_v}$ stands for the number of vehicles. Geometric sizes include wheelbase $\mathrm{L_w}$, front overhang length $\mathrm{L_f}$, rear overhang length $\mathrm{L_r}$, and car width $\mathrm{L_b}$ (**Fig. 2**). Mechanical constraints include

$$\begin{cases} |a_i(t)| \le \mathrm{a_{max}} \\ |v_i(t)| \le \mathrm{v_{max}} \\ |\phi_i(t)| \le \Phi_{\max} \\ |\omega_i(t)| \le \Omega_{\max} \end{cases}, \ t\in[0,t_{\mathrm{f}}],\ i=1,2,\ldots,\mathrm{N_v}, \tag{2}$$

where $\mathrm{a_{max}}$, $\mathrm{v_{max}}$, $\Phi_{\max}$, and $\Omega_{\max}$ represent boundaries on $a_i(t)$, $v_i(t)$, $\phi_i(t)$, and $\omega_i(t)$ respectively.

2.2 Collision avoidance

Collision avoidance plays a primary role in achieving motion feasibility. This subsection considers two types of collisions, namely, 1) vehicle-to-vehicle collisions, and 2) vehicle-to-barrier collisions. Suppose that each vehicle is rectangular. The requirement that two rectangles do not collide can be analytically described using our previously proposed triangle-area criterion [29–31], but the computational burden is overly heavy. Thus simplifications are made in this work. Specifically, two circles are utilized to cover the rectangular region of vehicle i (**Fig. 2**). Denoting the circle center closer to the rear wheals as $\mathrm{Cr}_i = \left(\mathrm{Cr}_{ix}, \mathrm{Cr}_{iy}\right)$ and the other $\mathrm{Cf}_i = \left(\mathrm{Cf}_{ix}, \mathrm{Cf}_{iy}\right)$, one can compute their locations via simple mathematics:

$$\begin{aligned}
&\left(\mathrm{Cr}_{ix}(t), \mathrm{Cr}_{iy}(t)\right) = \left(x_i(t) + \frac{\mathrm{L_w} + \mathrm{L_f} - 3\mathrm{L_r}}{4} \cdot \cos\theta_i(t),\ y_i(t) + \frac{\mathrm{L_w} + \mathrm{L_f} - 3\mathrm{L_r}}{4} \cdot \sin\theta_i(t) \right) \\
&\left(\mathrm{Cf}_{ix}(t), \mathrm{Cf}_{iy}(t)\right) = \left(x_i(t) + \frac{3\mathrm{L_w} + 3\mathrm{L_f} - \mathrm{L_r}}{4} \cdot \cos\theta_i(t),\ y_i(t) + \frac{3\mathrm{L_w} + 3\mathrm{L_f} - \mathrm{L_r}}{4} \cdot \sin\theta_i(t) \right), \\
&\qquad t \in [0, t_\mathrm{f}],\ i = 1, 2, \ldots, \mathrm{N_v}.
\end{aligned} \tag{3}$$

The radius R of either circle equals to $\sqrt{(\frac{\mathrm{L_r} + \mathrm{L_w} + \mathrm{L_f}}{4})^2 + (\frac{\mathrm{L_b}}{2})^2}$. Through this, collisions with vehicle i are sufficiently avoided if collisions to the aforementioned both circles are avoided. Constraints between each two of all the vehicles are described as:

$$\begin{cases}
\sqrt{\left(\mathrm{Cr}_{ix}(t) - \mathrm{Cr}_{jx}(t)\right)^2 + \left(\mathrm{Cr}_{iy}(t) - \mathrm{Cr}_{jy}(t)\right)^2} \ge 2\mathrm{R},\ \forall i, j = 1, \ldots, \mathrm{N_v},\ i \ne j \\
\sqrt{\left(\mathrm{Cr}_{ix}(t) - \mathrm{Cf}_{jx}(t)\right)^2 + \left(\mathrm{Cr}_{iy}(t) - \mathrm{Cf}_{jy}(t)\right)^2} \ge 2\mathrm{R},\ \forall i, j = 1, \ldots, \mathrm{N_v},\ i \ne j \\
\sqrt{\left(\mathrm{Cf}_{ix}(t) - \mathrm{Cf}_{jx}(t)\right)^2 + \left(\mathrm{Cf}_{iy}(t) - \mathrm{Cf}_{jy}(t)\right)^2} \ge 2\mathrm{R},\ \forall i, j = 1, \ldots, \mathrm{N_v},\ i \ne j
\end{cases},\ t \in [0, t_\mathrm{f}]. \tag{4}$$

In addition, collision between vehicle i and scenario barrier $y = \mathrm{BR}$ (**Fig. 2**) is avoided via

$$\begin{cases}
\mathrm{Cr}_{jy}(t) + \mathrm{R} \le \mathrm{BR} \\
\mathrm{Cf}_{jy}(t) + \mathrm{R} \le \mathrm{BR}
\end{cases},\ t \in [0, t_\mathrm{f}], \tag{5}$$

where BR is a parametric constant that determines the location of the scenario barrier.

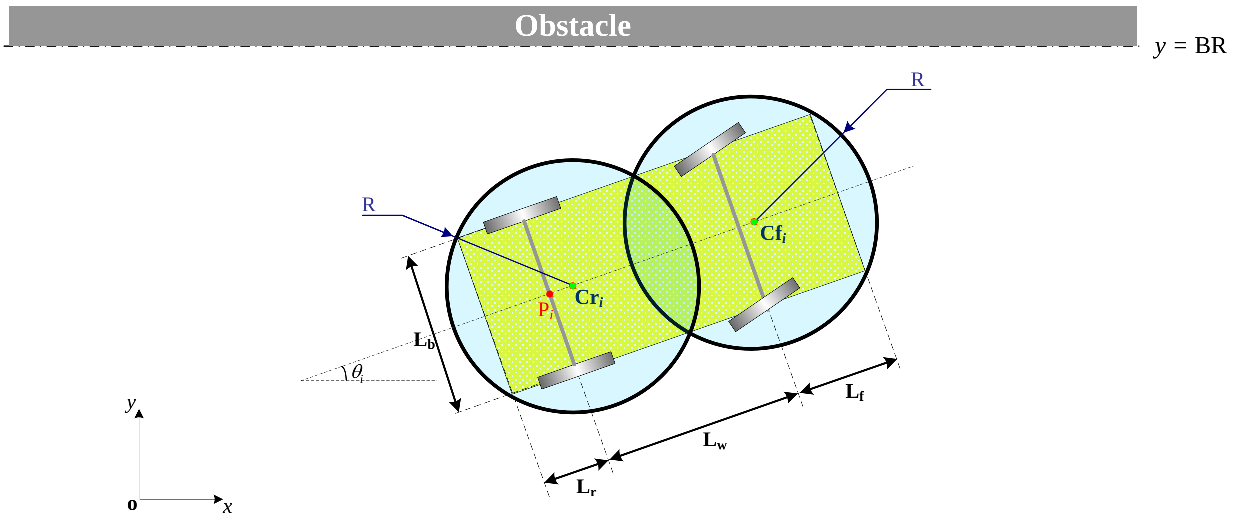


Fig. 2. Schematics on vehicle size, kinematics, and collision-avoidance constraints.

2.3 Scenario setup

Cooperative parking tasks are considered in this study. The configuration of each vehicle at the beginning is given as:

$$[x_i(0), y_i(0), \theta_i(0), v_i(0), \phi_i(0)] = [\bar{\mathrm{x}}_\mathrm{i}, \bar{\mathrm{y}}_\mathrm{i}, \bar{\theta}_\mathrm{i}, 0, 0],\ i = 1, 2, \ldots, \mathrm{N_v}, \tag{6}$$

where each $[\bar{\mathrm{x}}_\mathrm{i}, \bar{\mathrm{y}}_\mathrm{i}, \bar{\theta}_\mathrm{i}]$ is given *a priori*. During the entire parking process, vehicles are restricted to move between $y = \mathrm{AW}$ and $y = \mathrm{BW}$ (**Fig. 3**). These vehicles are expected to reach their target parking slots below $y = \mathrm{CW}$ (see the dashed boxes in **Fig. 3**) at the terminal moment. At $t = t_\mathrm{f}$, it is also required that

$$[v_i(t_\mathrm{f}), \phi_i(t_\mathrm{f})] = [0, 0],\ i = 1, 2, \ldots, \mathrm{N_v}. \tag{7}$$

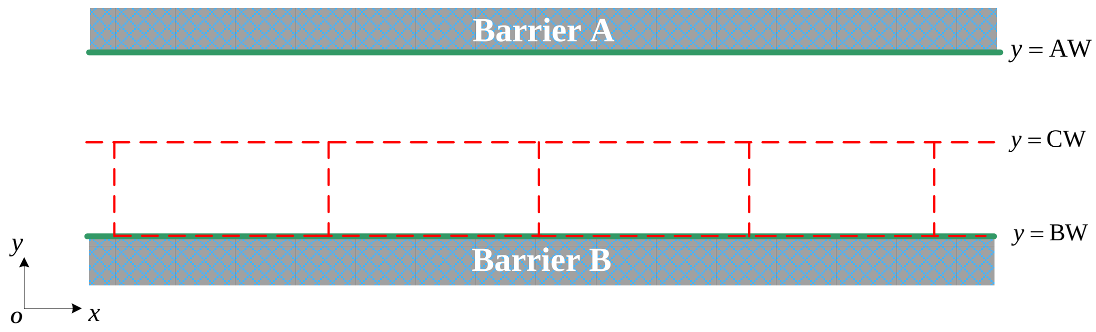


Fig. 3. Parametric notations related to vehicle size and kinematics.

All the aforementioned elements, together with an optimization objective, form an optimal control problem. The decision variables include $a_i(t)$, $\omega_i(t)$, and t_f ($t \in [0, t_\mathrm{f}]$, $i = 1, 2, \ldots, \mathrm{N_v}$). Parking completion time t_{f} is chosen as the minimization objective. Next section concerns about how to solve this minimum-time optimal control problem.

3. Optimal control problem solution

Orthogonal collocation direct transcription (OCDT) method is utilized to parameterize the aforementioned infinite-dimensional optimal control problem into a finite-dimensional nonlinear programming (NLP) problem [32,33]. A powerful NLP-solver, namely interior point method (IPM), is adopted [34]. However, directly applying OCDT+IPM is inefficient (simulation results reported in Section 4 will show this). That is because the large numbers of collision-avoidance constraints, when converted into parameterized forms, still exist in the NLP problem. This study introduces an initialization based methodology to ease solving that NLP problem. For the convenience of understanding, the NLP discretized directly from the original optimal control problem is referred to as the “original NLP problem” in this section.

3.1 Related works

Initialization is known to significantly influence the NLP-solving process, especially when gradient-based solvers are utilized [35]. Even, an initial guess alters the convergence result in some typical programming cases [36]. Herein, an initial guess refers to a roughly guessed solution from which an optimization process starts, and initialization stands for generation of initial guess. Häusler et al. [37] generated initial guess by connecting the desired initial and final positions of the vehicles via straight lines for motion planning problem. Such initial guesses are not feasible or near-feasible (because straight lines never guarantee trajectory feasibility, unless in simple cases), thus the difficulties in the NLP problem are not eased. Lee and Park [38] utilized genetic algorithm to generate initial guess in intersection management. However, the initial guess quality is suspicious because stochastic optimizers are inefficacious in handling highly constrained problems [39–45]. In a wider scope, i.e., in solving dynamic optimization problems, initialization strategies have been developed for stirred tank reactor control [35], tractor-trailer vehicle trajectory planning [36], spaceship orbit design [46], water-using network construction [47], power flow design [48], and single-vehicle parking trajectory planning [49]. However, these studies are either case-dependent or unable to ease the difficult part in the NLP problem we concern.

3.2 Progressively constrained dynamic optimization methodology

The main idea of our proposed initialization methodology is sequentially solving a series of simplified NLP subproblems with increasing complexity. Here, the simplifications are made through temporarily removing some of the complicated constraints from the original NLP problem. This implies that the first simplified subproblem removes the largest number of constraints from the original problem, while the last simplified subproblem removes the least number of constraints. To begin with, subproblem 1, the easiest subproblem, is solved, and the obtained solution is recorded as the initial guess. Thereafter, subproblem 2 is solved with the recorded initial guess, and the obtained solution updates the previously recorded initial guess. This process iteratively continues until all the constrained conditions in the original NLP problem are satisfied. In this study, the simplifications are related to the collision-avoidance constraints among the vehicles, because this type of constraints takes up a major part of the total NLP-solving difficulties. In addition, those simplified NLP subproblems are set as fixed-time, rather than free-time.

How to add the constraints progressively back towards the original NLP problem is a core highlight of this initialization methodology. A matrix $\mathbf{C} \in \mathbb{R}^{\mathrm{N_V} \times \mathrm{N_V}}$ is introduced to measure and record the collision degrees among each two vehicles. Suppose that $\mathrm{N_{fe}}$ finite elements are utilized in the OCDT method to describe the parameterized control/state profiles [32], the collision extent matrix $\mathbf{C}$ on the basis of any given solution can be evaluated by **Algorithm 1**. When $\mathbf{C}$ is available, the largest element of $\mathbf{C}$, which is supposed to be $c_{m,n}$, is found, and then collision-free restrictions (i.e., **Eq. (4)**) between the mth and nth vehicle is imposed on the subproblem solved the most recently so as to form the next subproblem. When the “next subproblem” is solved, the obtained solution is evaluated by **Algorithm 1** again and some constraints are imposed to form the next subproblem. This process repeats until every element of $\mathbf{C}$ equals zero, which means a thoroughly collision-free solution is obtained. Since the obtained solutions to those subproblems are gradually getting closer towards being a feasible solution to the original problem, thus we name this sequential computation framework as progressively constrained dynamic optimization (PCDO). Pseudo-code of the PCDO method is given in **Algorithm 2**.

Algorithm 1. Computation of collision extent matrix $\mathbf{C}$

```
1.  Load a to-be-evaluated solution;
2.  Set each element $c_{i,j}$ of $\mathbf{C} \in \mathbb{R}^{\mathrm{N_V} \times \mathrm{N_V}}$ to 0;
3.  For $i = 1 : \mathrm{N_{fe}}$, do
4.      For $ii = 1 : (\mathrm{N_V} - 1)$, do
5.          For $jj = (ii + 1) : \mathrm{N_V}$, do
6.              If $\sqrt{\left(\mathrm{Cr}_{iix} - \mathrm{Cr}_{jjx}\right)^2 + \left(\mathrm{Cr}_{iiy} - \mathrm{Cr}_{jjy}\right)^2} < 2\mathrm{R}$ or $\sqrt{\left(\mathrm{Cr}_{iix} - \mathrm{Cf}_{jjx}\right)^2 + \left(\mathrm{Cr}_{iiy} - \mathrm{Cf}_{jjy}\right)^2} < 2\mathrm{R}$ or $\sqrt{\left(\mathrm{Cf}_{iix} - \mathrm{Cf}_{jjx}\right)^2 + \left(\mathrm{Cf}_{iiy} - \mathrm{Cf}_{jjy}\right)^2} < 2\mathrm{R}$, then
7.                  Set $c_{ii,jj} = c_{ii,jj} + 1$;
8.              End if
9.          End for
10.     End for
11. End for
12. Store matrix $\mathbf{C}$ and exit.
```

Remark 1. The time complexity of Algorithm 1 is $O\left(\mathrm{N_{fe}} \cdot \mathrm{N_V^2}\right)$, since $\mathrm{N_{fe}}$ is the number of finite elements in the OCDT method, and the computation between steps 4 and 10 sum up to be $\mathrm{N_V}(\mathrm{N_V} - 1)/2$ times.

Algorithm 2. PCDO method

```
1.  Discard all the collision-avoidance constraints of the original NLP problem, fix $t_{\mathrm{f}}$ to $\mathrm{t}_{\xi}$, and
    name this simplified subproblem as $P_{current}$;
2.  Solve $P_{current}$;
3.  Record the obtained solution as the initial guess;
4.  Compute collision extent matrix $\mathbf{C}$ using Algorithm 1;
5.  While $\mathbf{C}$ is not a zero matrix, do
```

```
6.      Locate the largest one element of $\mathbf{C}$, and update $P_{current}$ by adding the corresponding
        collision-avoidance constraints (in the form of Eq. (4)) to it *;
7.      Solve $P_{current}$ with initial guess;
8.      Update the initial guess by the obtained solution;
9.      Update the collision extent matrix $\mathbf{C}$ using Algorithm 1;
10. End while
11. Store the current initial guess;
12. Exit.
```

* Some checkpoints are left off the description of Algorithm 2 for the sake of clarity: if there exist at least two elements in $\mathbf{C}$ with exactly the same largest value, then one of them is randomly selected in step 6.

Remark 2. The time complexity of Algorithm 2 is $O\left(\mathrm{N_{fe}} \cdot \mathrm{N_V^6} \cdot \log(\mathrm{N_V^2})\right)$. The most time consuming part of the algorithm occurs within the while loop. In the worst case, the while loop iterates for at most $\mathrm{N_V^2}$ times. In the while loop, sorting the largest elements takes $O\left(\mathrm{N_V^2} \cdot \log(\mathrm{N_V^2})\right)$, and Algorithm 1 takes $O\left(\mathrm{N_{fe}} \cdot \mathrm{N_V^2}\right)$.

Remark 3. Algorithm 2 outputs a feasible rather than optimal solution to the original problem, because t_{f} is still fixed.

3.3 Overall motion planning framework

When a feasible solution to the original NLP problem is available (calculated by the PCDO method), it serves as the initial guess to solve the original problem. Then the obtained collocation points render continuous profiles in the piecewise Lagrange polynomial structure, which form the optimal solution to the original optimal control problem. A diagram of the complete MVMP problem-solving procedures is depicted in **Fig. 4**.

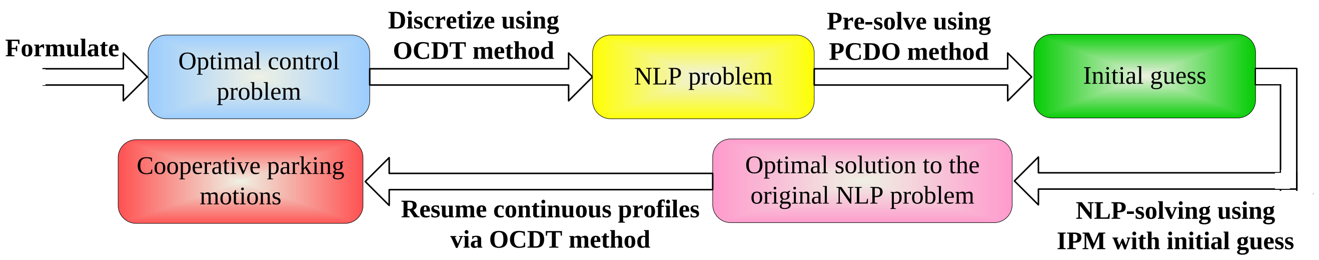

Formulate
Optimal control problem
Discretize using OCDT method
NLP problem
Pre-solve using PCDO method
Initial guess
NLP-solving using IPM with initial guess
Optimal solution to the original NLP problem
Resume continuous profiles via OCDT method
Cooperative parking motions

Fig. 4. Complete flowchart of the proposed MVMP methodology.

4. Simulations and discussions

Simulations were conducted in AMPL (A Mathematical Programming Language) environment [50] and executed on Intel Core 2 Duo CPU with 2GB RAM that runs at 2.53 GHz. Implementing the constraints in AMPL is straightforward, and requires minimum machine translation from our concerned formulation. IPOPT (version 3.12.4, a software package of IPM) [34] under default settings was utilized as the NLP solver.

Nominal parametric settings are listed in **Table 1**. The simulation results are shown in **Table 2** and **Figs. 5–11**. The optimized state/control profiles are depicted in **Fig. 12** when Case 1 is taken as the example. Recognizing that dynamic parking motions are difficult to imagine through static figures, we provide a video that contains all the simulation results at https://youtu.be/2J380cVRnSg.

Table 1. User-specific parametric settings.

Parameter	Description	Setting
K	Number of collocation points in each finite element in OCDT method	3
$\mathrm{N_{fe}}$	Number of finite element in OCDT method	20
$\mathrm{L_f}$	Vehicle front overhang length	0.960 m
$\mathrm{L_w}$	Vehicle wheelbase	2.800 m
$\mathrm{L_r}$	Vehicle rear overhang length	0.929 m
$\mathrm{L_b}$	Vehicle width	1.942 m
$\mathrm{N_v}$	Number of vehicles	4
$\mathrm{v_{max}}$	Upper bound of $\lvert v_i(t)\rvert,\ i=1,2,...,\mathrm{N_v}$	2.0 m/s
$\Phi_{\max}$	Upper bound of $\lvert \phi_i(t)\rvert,\ i=1,2,...,\mathrm{N_v}$	0.5435 rad
$\mathrm{a_{max}}$	Upper bound of $\lvert a_i(t)\rvert,\ i=1,2,...,\mathrm{N_v}$	0.5 m/s^2
$\omega_{\max}$	Upper bound of $\lvert \omega_i(t)\rvert,\ i=1,2,...,\mathrm{N_v}$	0.3294 rad/s

Table 2. Simulation case setups and results.

Case #	Description	Optimized t_f
1	$\begin{bmatrix}\bar{x}_1,\bar{y}_1,\bar{\theta}_1\\ \bar{x}_2,\bar{y}_2,\bar{\theta}_2\\ \bar{x}_3,\bar{y}_3,\bar{\theta}_3\\ \bar{x}_4,\bar{y}_4,\bar{\theta}_4\end{bmatrix}=\begin{bmatrix}-15 & -2.5 & 0\\ -8 & -3.5 & 0\\ 15 & -2 & 0\\ 8 & -3 & 0\end{bmatrix},\ \begin{cases}\mathrm{AW}=2.0\\ \mathrm{BW}=-6.0\\ \mathrm{CW}=-2.8\end{cases},\ \begin{cases}10\le x_1(t_\mathrm{f})\le 16\\ 16\le x_2(t_\mathrm{f})\le 22\\ -8.5\le x_3(t_\mathrm{f})\le -2.5\\ -14.5\le x_4(t_\mathrm{f})\le -8.5\end{cases},$ and $\mathrm{t}_\xi=20$	18.7258 s

2	$\begin{bmatrix} \bar{x}_1,\bar{y}_1,\bar{\theta}_1 \\ \bar{x}_2,\bar{y}_2,\bar{\theta}_2 \\ \bar{x}_3,\bar{y}_3,\bar{\theta}_3 \\ \bar{x}_4,\bar{y}_4,\bar{\theta}_4 \end{bmatrix} = \begin{bmatrix} -15 & 0 & 0 \\ -8 & -1 & 0 \\ 15 & 0.5 & 0 \\ 8 & -0.5 & 0 \end{bmatrix}$, $\begin{cases} AW = 3.0 \\ BW = -8.0, \\ CW = -2.0 \end{cases}$ $\begin{cases} 11.4 \le x_1(t_f) \le 14.6 \\ 17.4 \le x_2(t_f) \le 20.6 \\ -7.1 \le x_3(t_f) \le -3.9 \\ -13.1 \le x_4(t_f) \le -9.9 \end{cases}$, and $t_\xi = 25$	22.4719 s
3	Same with Case 1, except that $\begin{bmatrix} \bar{\theta}_1 \\ \bar{\theta}_2 \\ \bar{\theta}_3 \\ \bar{\theta}_4 \end{bmatrix} = \begin{bmatrix} 0.6049 \\ -0.1647 \\ -0.4430 \\ 0.0938 \end{bmatrix}$.	18.3214 s
4	Same with Case 2, except that $\begin{bmatrix} \bar{\theta}_1 \\ \bar{\theta}_2 \\ \bar{\theta}_3 \\ \bar{\theta}_4 \end{bmatrix} = \begin{bmatrix} 0.3143 \\ -0.0292 \\ 0.6006 \\ -0.4662 \end{bmatrix}$.	22.6433 s
5	Same with Case 1, except that the minimization objective is changed to $t_f + \lambda \cdot \int_0^{t_f} \left(\sum_{i=1}^{N_v} \omega_i^2(\tau) \right) d\tau$, where $\lambda = 1$.	19.1196 s
6	Same with Case 1, except that a static vehicle 5, which is fixed at $[\bar{x}_5, \bar{y}_5, \bar{\theta}_5] = [2.5, -4.1, 0.3]$, is added in the scenario.	33.3929 s
7	Same with Case 6 except that vehicle 5 can move, and $t_\xi = 35$	28.4189 s

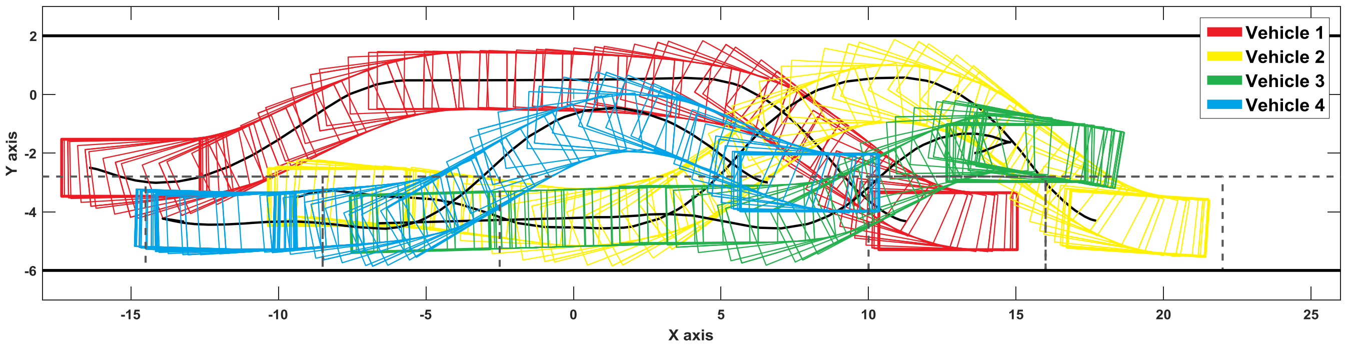


Fig. 5. Optimal parking motions in Case 1 ($t_f = 18.7258$ s).

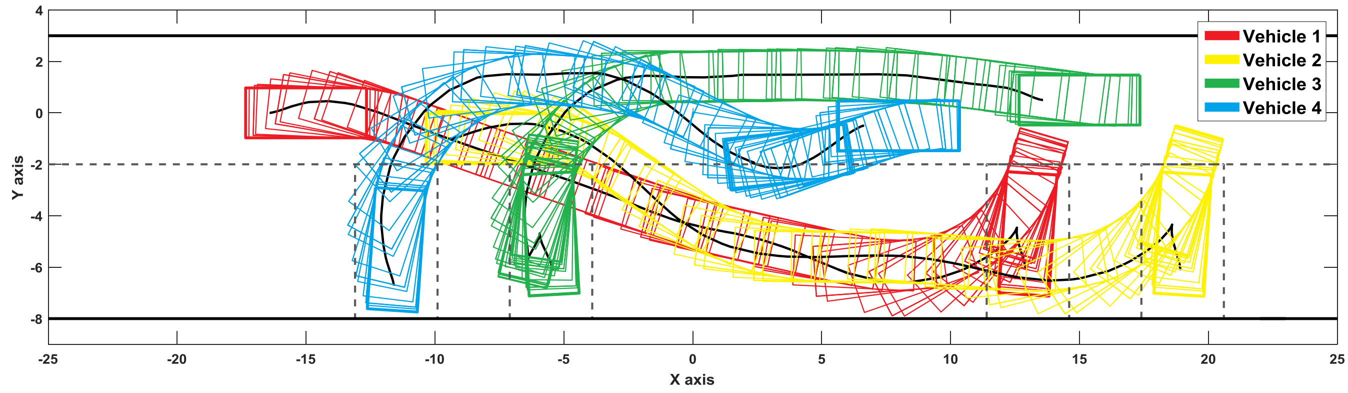


Fig. 6. Optimal parking motions in Case 2 ($t_f = 22.4719$ s).

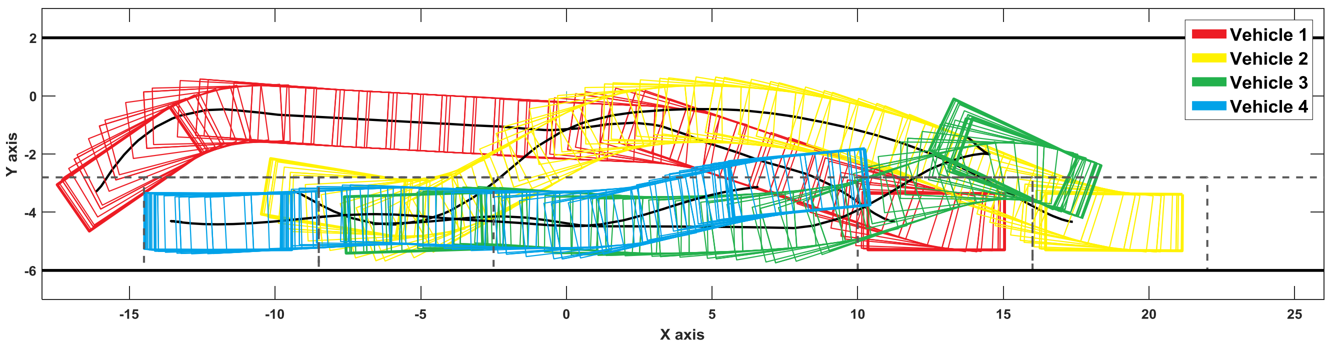


Fig. 7. Optimal parking motions in Case 3 ($t_f = 18.3214$ s).

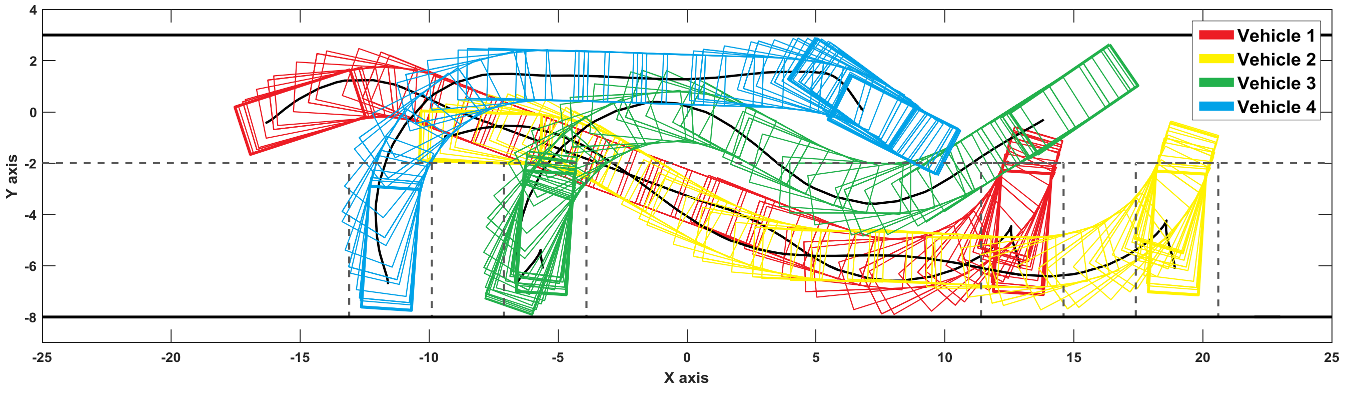


Fig. 8. Optimal parking motions in Case 4 ($t_f = 22.6433$ s).

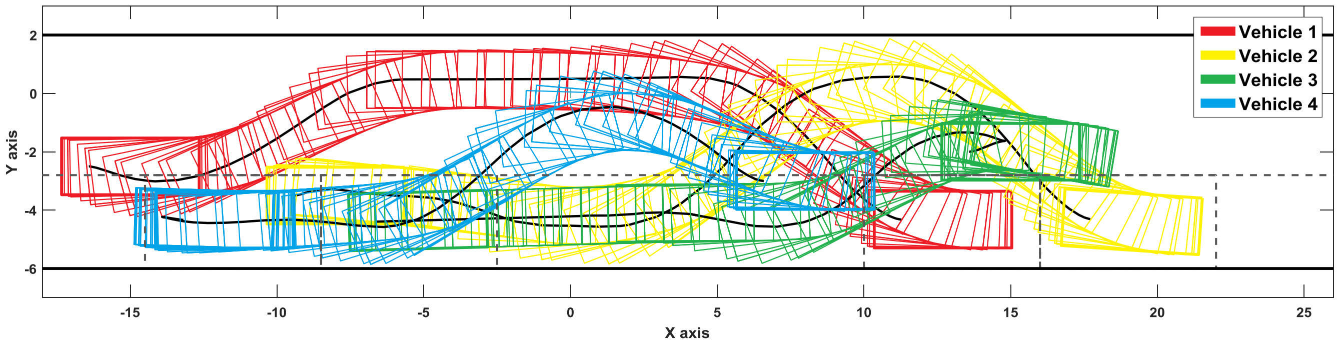


Fig. 9. Optimal parking motions in Case 5 ($t_f = 19.1196$ s).

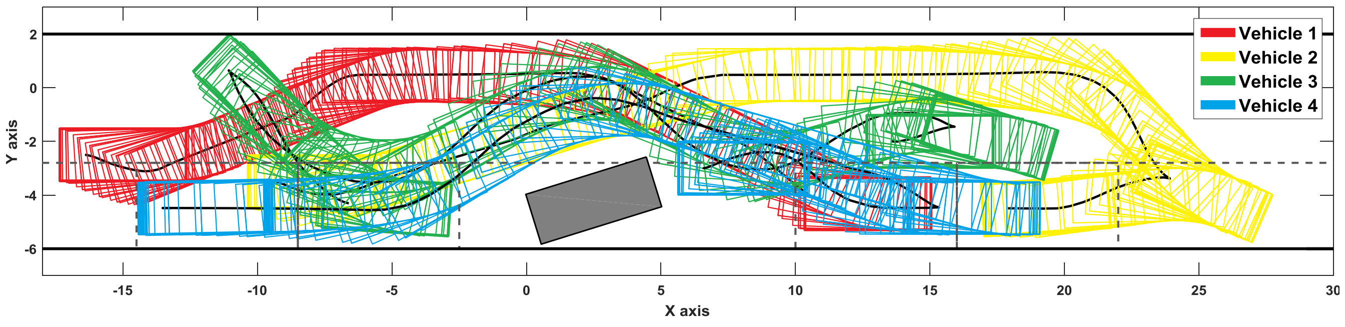


Fig. 10. Optimal parking motions in Case 6 ($t_f = 33.3929$ s).

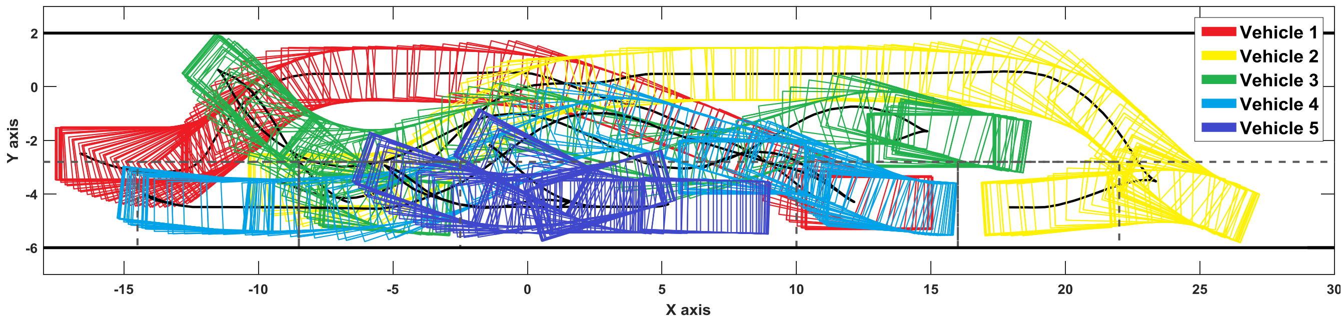
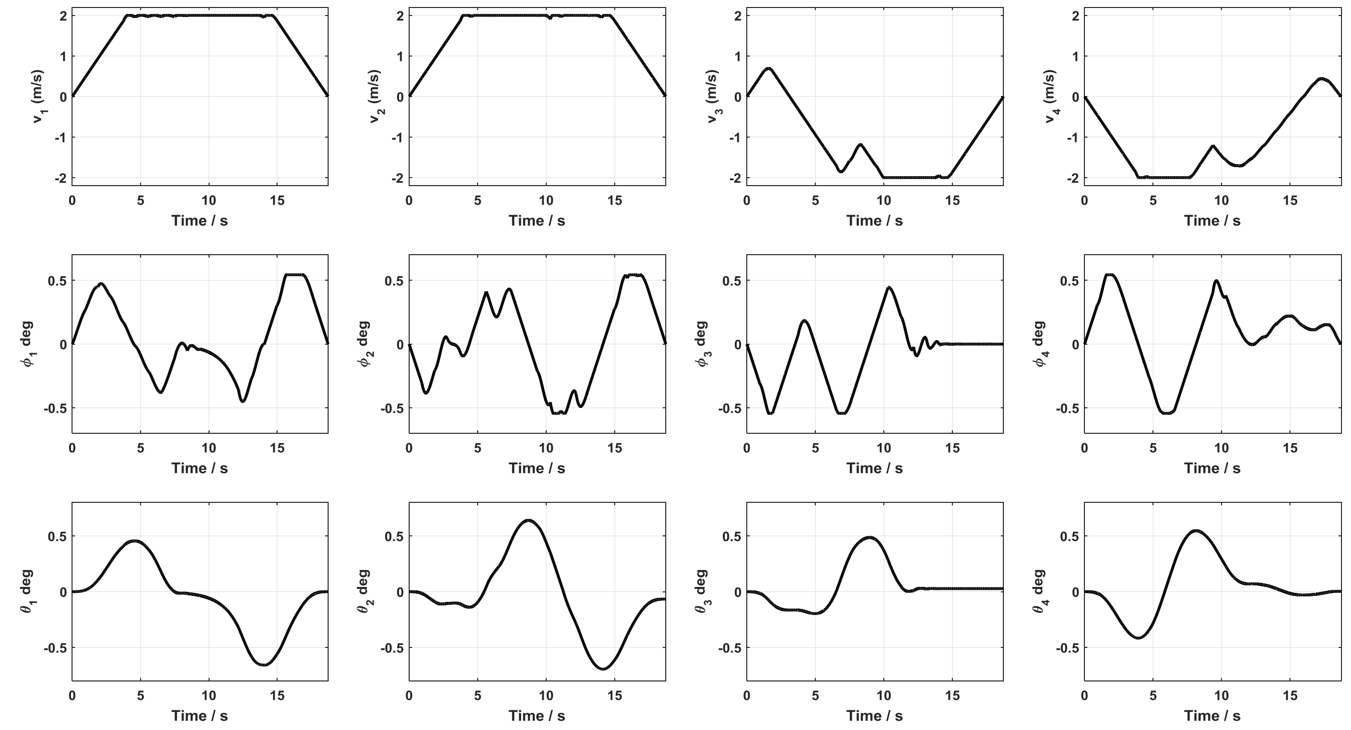
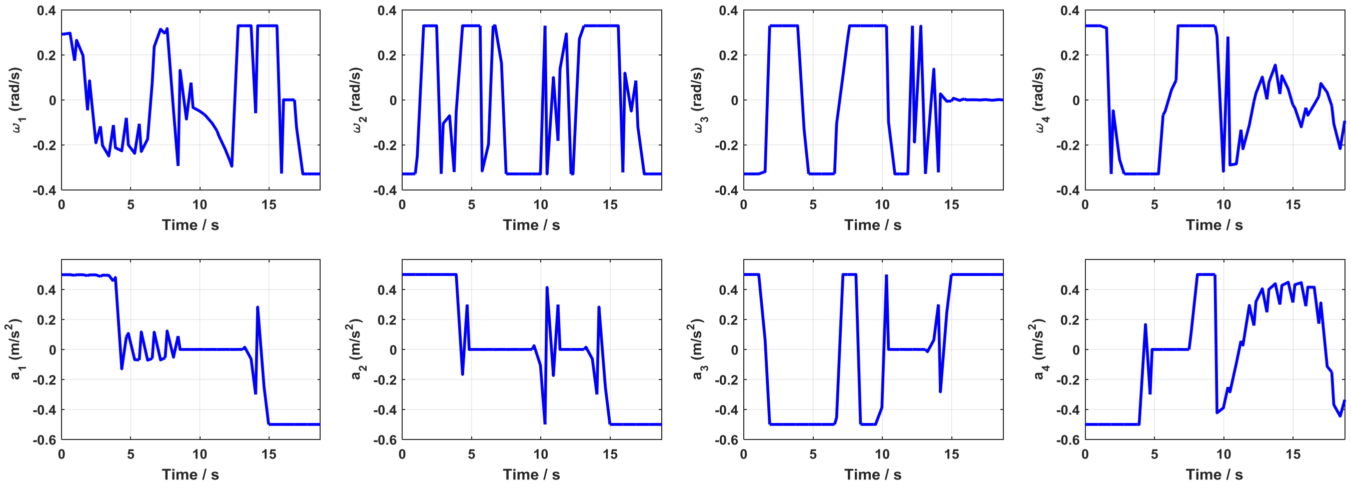


Fig. 11. Optimal parking motions in Case 7 ($t_f = 28.4189$ s).



(a)



(b)

Fig. 12. Optimal state/control profiles in Case 1: (a) state profiles, and (b) control profiles.

4.1 On the simulation results

This subsection discusses the optimized parking motions of Cases 1–7.

4.1.1 Comparison between Cases 1 and 3

Case 1 is a basic parallel parking case, while Case 3 makes differences in the randomly set starting orientations. In both cases, vehicles manage to reach their destinations successfully. Particularly, vehicle 3 slightly goes forward and then enters its own parking via backward motion directly in either case. That seems to be wasting time but actually not, because vehicle 3 adjusts its orientation and steering angles when going forward, so as to make the subsequent backward motion easier. More importantly, vehicle 3 would block the way of other vehicles and even cause congestions if it heads for the destination too soon. Since a congestion yields a larger t_{f}, vehicle 3 sacrifices slightly while all vehicles can complete the parking process earlier. It is worthwhile to emphasize that we do not impose any priority or coordination restrictions on the vehicles, and this observed phenomenon originates from a successful simultaneous computing process. The trajectories of other vehicles can be analyzed in a same way. As a brief summary, comparison between Cases 1 and 3 shows that 1) the proposed MVMP method is consistent in handling different cases, and 2) it is feasible to handle MVMP problems without assigning priorities and/or other constraints if simultaneous computation method is used.

4.1.2 Comparison between Cases 2 and 4

Case 2 is a basic vertical parking case, and Case 4 makes differences in the randomly set starting orientations. In Case 2, vehicle 3 stays on the left side of vehicle 4 at the very beginning, during which vehicle 4 moves slowly to give way to vehicle 3; thereafter, vehicle 4 goes farther than vehicle 3 in the *y* axis direction. By contrast in Case 4, vehicle 4 stays higher in the *y* axis direction than vehicle 3. Besides that, the evading trajectory of vehicle 3 (to avoid collisions with vehicles 1 and 2) is different. Changes in the initial configuration settings account for the differences in both cases. These differences are hard to propose by human experiences, but intuitively reasonable as they come out. This indicates that subjective knowledge based logics, principles, or rules are not efficient in harsh motion planning schemes; instead, computing with a complete and objective formulation is able to generate solutions uniformly.

4.1.2 Comparison between Cases 1 and 5

Case 5 is designed to evaluate the optimization performance when more than one criterion is considered (in the form of a weighted sum). Specifically in Case 5, the minimization objective is temporarily changed as $t_{\mathrm{f}} + \lambda \cdot \int_0^{t_{\mathrm{f}}} \left(\sum_{i=1}^{\mathrm{N_v}} \omega_i^2(\tau) \right) \mathrm{d}\tau$, wherein the newly added item accumulates the vehicle steering energy and λ is a user-specified weight parameter. Since an integral cannot directly serve as the optimization objective, a state variable $energy(t)$, together with the following equalities, should be incorporated in the original optimization problem:

$$\begin{cases} \dfrac{\mathrm{d}energy(t)}{\mathrm{d}t} = \sum_{i=1}^{\mathrm{N_v}} \omega_i^2(t) \\ energy(0) = 0 \end{cases}. \qquad (8)$$

Through this, the minimization objective now becomes $t_{\mathrm{f}} + \lambda \cdot energy(t_{\mathrm{f}})$, which is regarded as a time-energy criterion in previous studies [51]. How λ affects the optimization result is studied via a series of simulations in **Table 3**. As λ grows, $energy(t_{\mathrm{f}})$ decreases, and t_{f} increases, which is consistent with common practice. **Fig. 13** depicts the related profiles in Case 5 with $\lambda = 1$. Compared with those in **Fig. 12**, the obtained ω_i variables in **Fig. 13** are smoother, because unnecessary oscillations in the optimized ω_i of Case 1 are against the new minimization objective in Case 5.

Table 3. Various parameter settings of λ and results.

λ	t_{f} index	$energy(t_{\mathrm{f}})$ index
0	18.7258	4.9748
0.00001	18.7258	4.7198
0.0001	18.7258	4.6533
0.001	18.7260	4.2538
0.01	18.7306	3.0823
0.1	18.7553	2.2806
1	19.1196	1.3878

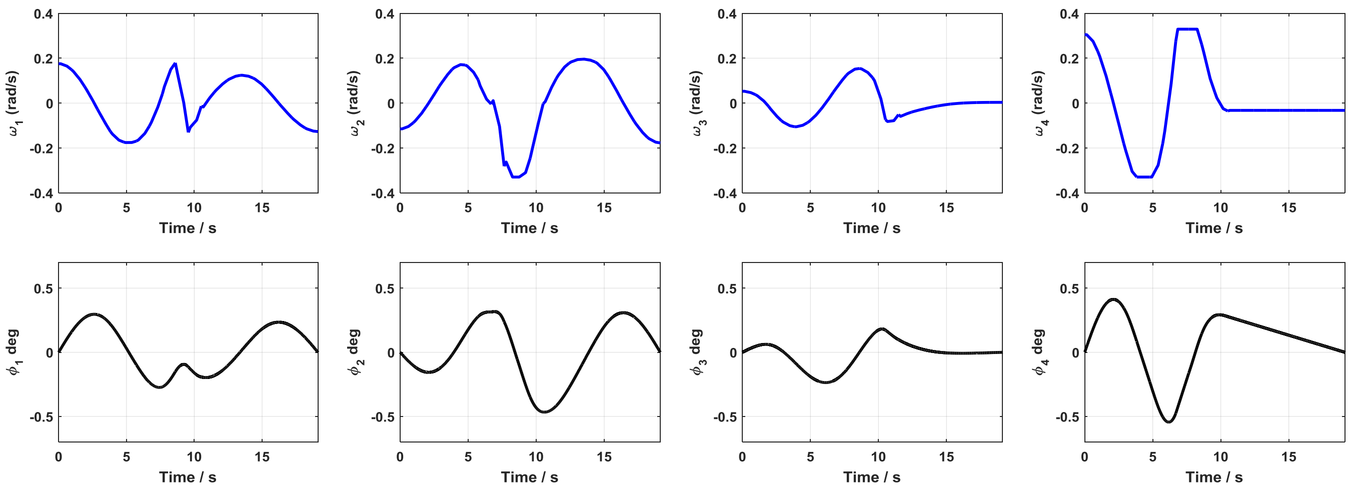


Fig. 13. Part of optimal state/control profiles in Case 5 ($\lambda = 1$).

4.1.3 Comparison between Cases 1 and 6

Case 6 differs from Case 1 in adding a static vehicle. In order to avoid that obstacle, cooperative trajectories of the four vehicles are far more complicated than those in Case 1. Specifically, vehicles 3 and 4 locally adjust so as to give way to vehicle 2; vehicle 1 moves slowly at the beginning to give way to vehicles 3 and 4 when they encounter at the bottleneck region $x \in [0,5]$; at last, vehicle 3 waits for some time until its target spot is left vacant by vehicle 4.

4.1.4 Comparison between Cases 6 and 7

Compared with Case 6, Case 7 allows the static vehicle move, thus it can be regarded as vehicle 5, which is not heading for any parking spot. In this case, the only mission of vehicle 5 is to evade other vehicles so as to ease the entire parking process. Although the cooperative trajectories of vehicles 1–4 are similar to those in Case 6, the optimization objective t_f decreases by 14.89%. This shows that thorough cooperation helps to save time.

In addition to that simple conclusion, Case 7 provides further indications if it is understood in the following way. Suppose there are 5 vehicles, they start from where they start in Case 7; vehicles 1–4 aim to stop in specified parking spots; and vehicle 5 aims to stop at where it stops in the optimized results of Case 7. A typical joint computing strategy is letting vehicle 5 remain still while vehicles 1–4 move, and then letting vehicle 5 move when the other 4 vehicles have already reached their destinations.

An alternative strategy is letting all the vehicles move all at once, that is, the simultaneous strategy. Consequently, the aforementioned joint computing strategy would take $(33.3929+\xi)$ seconds, wherein $\xi>0$ denotes the time spent on motion planning of vehicle 5. In contrast, the simultaneous strategy takes 28.4189 seconds. In this sense, comparison between Cases 6 and 7 shows the advantage of simultaneous computation.

4.2 On the NLP-solving methodology

This subsection first reports the procedures of the proposed PCDO method, and then carries out further analyses.

4.2.1 Log of NLP-solving process

The log reported in **Table 4** helps readers understand the principle of PCDO and the NLP-solving process. Case 1 is taken as an example. Note that the step-by-step initialization procedures are also recorded in the aforementioned video link.

Table 4. Log of initialization procedures in solving the NLP problem of Case 1.

Step #	Actions	Results	Computation time	Problem scale	
				# of decision variables	# of constraints
1	Set $t_\xi=20$, solve an NLP problem with all the multi-vehicle collision avoidance constraints discarded, compute $\mathbf{C}$, and store the solution as initial guess.	$t_f=20$, $\mathbf{C}=\begin{bmatrix}0&22&20&21\\0&0&17&20\\0&0&0&37\\0&0&0&0\end{bmatrix}$.	27.643 s	2844	4764
2	Add collision avoidance constraints between **vehicles 3 and 4** to the aforementioned NLP problem, solve it with initial guess, compute $\mathbf{C}$, and store the solution as initial guess.	$t_f=20$, $\mathbf{C}=\begin{bmatrix}0&0&12&12\\0&0&12&12\\0&0&0&0\\0&0&0&0\end{bmatrix}$.	0.936 s	2844	5080
3	Add collision avoidance constraints between **vehicles 1 and 4** to the aforementioned NLP problem, solve it with initial guess, compute $\mathbf{C}$, and store the solution as initial guess.	$t_f=20$, $\mathbf{C}=\begin{bmatrix}0&0&12&0\\0&0&12&10\\0&0&0&0\\0&0&0&0\end{bmatrix}$.	0.624 s	2844	5396
4	Add collision avoidance constraints between **vehicles 1 and 3** to the aforementioned NLP problem, solve it with initial guess, compute $\mathbf{C}$, and	$t_f=20$, $\mathbf{C}=\begin{bmatrix}0&0&0&0\\0&0&11&12\\0&0&0&0\\0&0&0&0\end{bmatrix}$.	0.843 s	2844	5712

	store the solution as initial guess.				
5	Add collision avoidance constraints between **vehicles 2 and 4** to the aforementioned NLP problem, solve it with initial guess, compute **C**, and store the solution as initial guess.	$t_{\mathrm{f}} = 20$, $\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 11 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$.	1.123 s	2844	6028
6	Add collision avoidance constraints between **vehicles 2 and 3** to the aforementioned NLP problem, solve it with initial guess, compute **C**, and store the solution as initial guess.	$t_{\mathrm{f}} = 20$, $\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$.	1.092 s	2844	6344
7	Solve the original NLP problem with initial guess.	$t_{\mathrm{f}} = 18.7258$.	13.850 s	2845	6660

The progressive constrained initialization begins from Step 1 and ends in Step 6. During that period, the obtained t_{f} is fixed to 20, which means that, as mentioned in Remark 3, feasible rather than optimal solutions are expected. Necessity to fix $t_{\mathrm{f}} = \mathrm{t}_{\xi}$ is shown in the next subsection.

Comparing Steps 6 and 7, one may notice that when all collisions are avoided, i.e., when **C** becomes a zero matrix, the concerned NLP problem is still easier than the original problem, because collision avoidance between vehicles 1 and 2 are not yet restricted! This phenomenon is reasonable because not all vehicles in a vehicle team actually have collision chances. That implies that our proposed initialization method is able to facilitate the computation by discarding unnecessary constraints.

More importantly, one may notice that the results obtained in Steps 6 and 7 are quite different: vehicle 4 passes by vehicle 2 from its right-hand side as in Step 6, and from the right-hand side in Step 7 (see the provided video). The gap between Steps 6 and 7 shows more than just the difference between a feasible solution and an optimal solution, it demonstrates that simultaneous computation method is efficacious to promote solution quality (by thorough cooperation) even if it starts from a jointly computed initial guess.

4.2.2 On the PCDO method

Comparative simulations are conducted to verify the efficiency of the PCDO method.

Comparisons are considered in two ways: 1) whether it makes sense to simplify the pre-solving problems by fixing t_f so that only feasible solutions are obtained; and 2) whether other sequences/pathways to add the collision-avoidance constraints back to the original NLP problem make sense. Specifically, the following initialization algorithms are defined. The results are listed in **Table 5**.

Algorithm 3: same with Algorithm 2, expect that t_f is set as the minimization objective in the PCDO method.
Algorithm 4: same with Algorithm 2, expect that collision-constraints related to the two (rather than one) largest elements in $\mathbf{C}$ are added to $P_{current}$ in each loop.
Algorithm 5: same with Algorithm 2, expect that collision-constraints related to the all non-zero elements in $\mathbf{C}$ are added to $P_{current}$ all at once.

Table 5. Comparative experiments on variants of the initialization method.

Case #	Computation time (s)				
	Algorithm 2	**Algorithm 3**	**Algorithm 4**	**Algorithm 5**	**Without initialization**
1	46.111	450.678	166.959	281.861	failed *
2	69.566	498.289	174.206	failed	failed
3	83.945	248.240	failed	failed	failed
4	268.557	508.171	1410.701	failed	failed
5	324.245	806.483	1118.171	1206.869	failed
6	1215.544	1457.580	failed	failed	failed
7	1411.223	1705.419	failed	failed	failed

* A failure means either a converged infeasible solution or a non-converged solution after 3,000 iterations, according to the default settings of IPOPT [34].

These optimization results show that 1) initialization is a necessary part in solving complicated NLP problems; 2) temporarily fixing t_f during the pre-solving steps saves computational time; and 3) adding back collision-avoidance constraints too hastily at one time (i.e., Algorithms 4 and 5) renders more computational efforts, and even failures in challenging cases.

4.3 Comparison between simultaneous and joint strategies

This subsection provides a comparison between simultaneous and joint computation strategies in solving the aforementioned MVMP cases. Three typical joint computation strategies are applied. Strategy 1 divides the whole multi-vehicle team into subgroups, strategy 2 sequentially plans motion for each vehicle, and strategy 3 considers path planning + coordination. However, their abilities are suspicious when the MVMP scheme is harsh: if the destination of one vehicle is occupied by another vehicle at $t = 0$ (e.g., Cases 1, 3, 5, 6, and 7), then none of the three joint computation strategies is efficient.

5. Conclusions

This paper has introduced an initialization framework to ease the burdens of simultaneous computation methodologies when they are adopted to solve the MVMP problem. Simulation results validate that 1) initialization is necessary in solving such complicated NLP problems, and 2) designs in the PCDO method make sense, and 3) simultaneous computation is superior to typical joint computation methods. This work emphasizes the importance of using simultaneous computation methods, which provide high-quality solutions to cooperation related problems. For simultaneous computation, the current bottleneck is how to promote the real-time computation performance, which is actually our next work [52].

Acknowledgements

The authors are grateful to both reviewers for their comments and suggestions. This work is supported by the 973 Program of China under Grant 2012CB720503, National Nature Science Foundation of China under Grant 61374167 and 61573282, College Students' Science & Technology Innovation Program of Zhejiang Province (Xinmiao Talent Project) under Grant 2016R401239, and International Cooperative Research and Communication Funding Scholarship for Zhejiang University Doctoral Students under Grant 2016Y188310. Bai Li thanks Ms. Zengyan Liu for the inspiration during this study.

References

[1] K. Solovey, and D. Halperin, k-Color multi-robot motion planning, *International Journal of Robotics Research*. 33(1) (2014) 82–97.

[2] J. Ma, X. Li, F. Zhou, J. Hu, and B. B. Park, Parsimonious shooting heuristic for trajectory design of connected automated traffic part II: computational issues and optimization, *Transportation Research Part B: Methodological*. 95(2017) 421–441.

[3] K. S. Senthilkumar, and K. K. Bharadwaj, Multi-robot exploration and terrain coverage in an unknown environment, *Robotics and Autonomous Systems*. 60(1) (2012) 123–132.

[4] S. C. Ho, and W. Y. Szeto, A hybrid large neighborhood search for the static multi-vehicle bike-repositioning problem, *Transportation Research Part B: Methodological*. 95(2017) 340–363.

[5] I. A. Ntousakis, I. K. Nikolos, and M. Papageorgiou, Optimal vehicle trajectory planning in the context of cooperative merging on highways, *Transportation Research Part C: Emerging Technologies*. 71 (2016) 464–488.

[6] D. Bevly, X. Cao, M. Gordon, G. Ozbilgin, D. Kari, B. Nelson, J. Woodruff, M. Barth, C. Murray, A. Kurt, K. Redmill, and U. Ozguner, Lane change and merge maneuvers for connected and automated vehicles: a survey. *IEEE Transactions on Intelligent Vehicles*. 1(1) (2016) 105–120.

[7] S. J. Buckley, Fast motion planning for multiple moving robots. In *1989 IEEE International Conference on Robotics and Automation*, pp. 322–326, 1989.

[8] S. Liu, D. Sun, and C. Zhu, A dynamic priority based path planning for cooperation of multiple mobile robots in formation forming, *Robotics and Computer-Integrated Manufacturing*. 30(6) (2014) 589–596.

[9] M. G. Plessen, D. Bernardini, H. Esen, and A. Bemporad, Multi-automated vehicle coordination using decoupled prioritized path planning for multi-lane one-and bi-directional traffic flow control. In 2*016 IEEE 55th Conference on Decision and Control*, pp. 1582–1588, 2016.

[10] M. Shanmugavel, A. Tsourdos, B. White, and R. Żbikowski, Co-operative path planning of multiple UAVs using Dubins paths with clothoid arcs, *Control Engineering Practice*. 18(9) (2010) 1084–1092.

[11] F. L. Lian, and R. Murray, Real-time trajectory generation for the cooperative path planning of multi-vehicle systems. In *41st IEEE Conference on Decision and Control*, vol. 4, pp. 3766–3769, 2002.

[12] G. Wagner, and H. Choset, Subdimensional expansion for multirobot path planning, *Artificial Intelligence*. 219 (2015) 1–24.

[13] K. Solovey, O. Salzman, and D. Halperin, Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning, *International Journal of Robotics Research*. 35(5) (2016) 501–513.

[14] P. Yao, H. Wang, and Z. Su, Cooperative path planning with applications to target tracking and obstacle avoidance for multi-UAVs, *Aerospace Science and Technology*. 54 (2016) 10–22.

[15] K. Jose, and D. K. Pratihar, Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods, *Robotics and*

Autonomous Systems. 80 (2016) 34–42.

[16] P. Urcola, M. T. Lázaro, J. A. Castellanos, and L. Montano, Cooperative minimum expected length planning for robot formations in stochastic maps, *Robotics and Autonomous Systems*. 87 (2017) 38–50.

[17] J. S. Bellingham, M. Tillerson, M. Alighanbari, and J. P. How, Cooperative path planning for multiple UAVs in dynamic and uncertain environments. In *41st IEEE Conference on Decision and Control*, vol. 3, pp. 2816–2822, 2002.

[18] P. K. Das, H. S. Behera, and B. K. Panigrahi, A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning, *Swarm and Evolutionary Computation*. 28 (2016) 14–28.

[19] H. Shorakaei, M. Vahdani, B. Imani, and A. Gholami, Optimal cooperative path planning of unmanned aerial vehicles by a parallel genetic algorithm, *Robotica*. 34(4) (2016) 823–836.

[20] P. Abichandani, K. Mallory, and M. Y. Hsieh, Experimental multi-vehicle path coordination under communication connectivity constraints. In *Experimental Robotics* 2013 (pp. 183–197). Springer International Publishing.

[21] A. J. Häusler, A. Saccon, A. P. Aguiar, J. Hauser, and A. M. Pascoal, Cooperative motion planning for multiple autonomous marine vehicles, *IFAC Proceedings Volumes*. 45(27) (2012) 244–249.

[22] A. Richards, and J. P. How, Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *2002 American Control Conference*, vol. 3, pp. 1936–1941, 2002.

[23] F. Borrelli, D. Subramanian, A. U. Raghunathan, and L. T. Biegler, MILP and NLP techniques for centralized trajectory planning of multiple unmanned air vehicles. In *2006 American Control Conference*, pp. 5763–5768, 2006.

[24] P. Abichandani, G. Ford, H. Y. Benson, and M. Kam, Mathematical programming for multi-vehicle motion planning problems. In *2012 IEEE International Conference on Robotics and Automation*, pp. 3315–3322, 2012.

[25] J. Chakraborty, A. Konar, L. C. Jain, and U. K. Chakraborty, Cooperative multi-robot path planning using differential evolution, *Journal of Intelligent & Fuzzy Systems*. 20(1-2) (2009) 13–27.

[26] C. Tam, and R. Bucknall, Cooperative path planning algorithm for marine surface vessels. *Ocean Engineering*. 57 (2013) 25–33.

[27] M. Defoort, A. Kokosy, T. Floquet, W. Perruquetti, and J. Palos, Motion planning for cooperative unicycle-type mobile robots with limited sensing ranges: a distributed receding horizon approach, *Robotics and Autonomous Systems*. 57(11) (2009) 1094–1106.

[28] B. Li, H. Liu, D. Xiao, G. Yu, and Y. M. Zhang, Centralized and optimal motion planning for large-scale AGV systems: A generic approach, *Advances in Engineering Software*. 106(2017) 33–46.

[29] B. Li, Y. M. Zhang, and Z. Shao, Spatio-temporal decomposition: a knowledge-based initialization strategy for parallel parking motion optimization, *Knowledge-Based Systems*. 107 (2016) 179–196.

[30] B. Li, K. Wang, and Z. Shao, Time-optimal trajectory planning for tractor-trailer vehicles via simultaneous dynamic optimization. In *2015 IEEE/RSJ International Conference on Intelligent*

Robots and Systems. pp. 3844–3849, 2015.

[31] B. Li, and Z. Shao, A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles, *Knowledge-Based Systems*, 86 (2015) 11–20.

[32] A. Cervantes, and L. T. Biegler, Large-scale DAE optimization using a simultaneous NLP formulation, *AIChE Journal*. 44(5) (1998) 1038–1050.

[33] B. Li, and Z. Shao, Simultaneous dynamic optimization: A trajectory planning method for nonholonomic car-like robots, *Advances in Engineering Software*. 87 (2015) 30–42.

[34] A. Wächter, and L. T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, *Mathematical Programming*. 106(1) (2006) 25–57.

[35] S. M. Safdarnejad, J. D. Hedengren, N. R. Lewis, and E. L. Haseltine, Initialization strategies for optimization of dynamic systems, *Computers & Chemical Engineering*. 78 (2015) 39–50.

[36] B. Li, and Z. Shao, An incremental strategy for tractor-trailer vehicle global trajectory optimization in the presence of obstacles. In *2015 IEEE International Conference on Robotics and Biomimetics*, pp. 1447–1452, 2015.

[37] A. J. Häusler, A. Saccon, A. P. Aguiar, J. Hauser, and A. M. Pascoal, Energy-optimal motion planning for multiple robotic vehicles with collision avoidance, *IEEE Transactions on Control Systems Technology*. 24(3) (2016) 867–883.

[38] J. Lee, and B. Park, Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment, *IEEE Transactions on Intelligent Transportation Systems*. 13(1) (2012) 81–90.

[39] B. Liu, S. Koziel, and Q. Zhang, A multi-fidelity surrogate-model-assisted evolutionary algorithm for computationally expensive optimization problems, *Journal of Computational Science*. 12 (2016) 28–37.

[40] B. Li, and Z. Shao, Precise trajectory optimization for articulated wheeled vehicles in cluttered environments, *Advances in Engineering Software*. 92 (2016) 40–47.

[41] S. Kaboli, J. Selvaraj, and N. Rahim, Rain-fall optimization algorithm: a population based algorithm for solving constrained optimization problems, *Journal of Computational Science*. 19 (2017) 31–42.

[42] B. Li, R. Chiong, and M. Lin, A two-layer optimization framework for UAV path planning with interval uncertainties, In *2014 IEEE Symposium on Computational Intelligence in Production and Logistics Systems (CIPLS)*. (2014) 120–127.

[43] T. Ngo, A. Sadollah, and J. Kim, A cooperative particle swarm optimizer with stochastic movements for computationally expensive numerical optimization problems, *Journal of Computational Science*. 13 (2016) 68–82.

[44] B. Li, R. Chiong, and L. Gong, Search-evasion path planning for submarines using the artificial bee colony algorithm, In *2014 IEEE Congress on Evolutionary Computation (CEC)*. (2014) 528–535.

[45] A. Yadav, and K. Deep, An efficient co-swarm particle swarm optimization for non-linear

constrained optimization, *Journal of Computational Science*. 5(2) (2014) 258–268.

[46] K. Subbarao, and B. M. Shippey, Hybrid genetic algorithm collocation method for trajectory optimization, *Journal of Guidance, Control, and Dynamics*. 32(4) (2009) 1396–1403.

[47] B. H. Li, and C. T. Chang, A simple and efficient initialization strategy for optimizing water-using network designs, *Industrial & Engineering Chemistry Research*. 46(25) (2007) 8781–8786.

[48] Y. C. Wu, and A. S. Debs, Initialisation, decoupling, hot start, and warm start in direct nonlinear interior point algorithm for optimal power flows, *IEE Proceedings-Generation, Transmission and Distribution*. 148(1) (2001) 67–75.

[49] B. Li, K. Wang, and Z. Shao, Time-optimal maneuver planning in automatic parallel parking using a simultaneous dynamic optimization approach, *IEEE Transactions on Intelligent Transportation Systems*. 17(11) (2016) 3263–3274.

[50] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*. San Francisco, CA, USA: Scientific, 2003.

[51] Z. Shiller, Time-energy optimal control of articulated systems with geometric path constraints. In *1994 IEEE International Conference on Robotics and Automation*, pp. 2680–2685, 1994.

[52] B. Li, Y. Feng, Y. Zhang, Y. Ge, Z. Shao, and H. X. Liu, Balancing computation speed and quality: a decentralized motion planner for cooperative lane changes of connected and automated vehicles, *Knowledge-Based Systems*, submitted.

Bai Li received his B.S. degree in 2013 from the School of Advanced Engineering, Beihang University (previously known as Beijing University of Aeronautics and Astronautics), China. From 2013 till now, he has been working toward his Ph.D. degree in the College of Control Science and Engineering, Zhejiang University, China. His research interests focus on the computational issues related to motion planning of connected and automated vehicles.

Youmin Zhang is a Professor in the Department of Mechanical and Industrial Engineering at Concordia University, Montreal, Canada. He received his Ph.D., M.S., and B.S. degrees from the Department of Automatic Control, Northwestern Polytechnical University, Xian, China, in 1995, 1986, and 1983, respectively. He held also several teaching and research positions previously in Northwestern Polytechnical University, University of New Orleans, Louisiana State University, State University of New York at Binghamton, The University of Western Ontario, and Aalborg University, respectively. He has published 4 books, over 420 journal and conference papers, and book chapters.

Zhijiang Shao received his B.S. degree in 1992 from the Department of Chemical Engineering, Zhejiang University, China, and Ph.D. degree in 1997 from the College of Control Science and Engineering, Zhejiang University, China. He was a lecturer in Zhejiang University from 1997 to 1999, and an Associate Professor from 1999 to 2004. Since 2004, he has been a Professor in the said university. His research interests include large-scale optimization, real-time optimization, process system optimization, and dynamic optimization.

Ning Jia received his B.S. degree in 2005 from the Department of Management, Shandong University, China, and Ph.D. degree in 2010 from Institute of Systems Engineering, Tianjin University, China. Since 2010, he has been working in the Institute of Systems Engineering, Tianjin University. Now he is an associate professor in the College of Management and Economics in the above university. His research interests include intelligent transportation systems, and urban traffic control & management.