Optimal Cooperative Maneuver Planning for Multiple Nonholonomic Robots in a Tiny Environment via Adaptive-Scaling Constrained Optimization

Bai Li, Yakun Ouyang, Youmin Zhang, Tankut Acarman, Qi Kong, and Zhijiang Shao

***Abstract*—This letter is focused on the time-optimal Multi-Vehicle Trajectory Planning (MVTP) problem for multiple car-like robots when they travel in a tiny indoor scenario occupied by static obstacles. Herein, the complexity of the concerned MVTP task includes i) the non-convexity and narrowness of the environment, ii) the nonholonomy and nonlinearity of the vehicle kinematics, iii) the pursuit for a time-optimal solution, and iv) the absence of predefined homotopic routes for the vehicles. The aforementioned factors, when mixed together, are beyond the capability of the prevalent coupled or decoupled MVTP methods. This work proposes an adaptive-scaling constrained optimization (ASCO) approach, aiming to find the optimum of the nominally intractable MVTP problem in a decoupled way. Concretely, an iterative computation framework is built, wherein each intermediate subproblem contains only risky collision avoidance constraints within a certain range, thus being tractable in the scale. During the iteration, the constraint activation scale can change adaptively, thereby enabling to promote the convergence rate, to recover from an intermediate failure, and to get rid of a poor initial guess. ASCO is compared versus the state-of-the-art MVTP methods and is validated in real experiments conducted by a team of three car-like robots.**

***Index Terms*—Path planning for multiple mobile robots or agents, nonholonomic motion planning, optimization and optimal control, motion and path planning.**

Manuscript received October 13, 2020; accepted January 20, 2021. Date of publication February 2, 2021; date of current version March 1, 2021. This letter was recommended for publication by Associate Editor D. Saldaña and Editor M. A. Hsieh upon evaluation of the reviewers' comments. This work was supported in part by Galatasaray University Scientific Research Support under Grant #19.401.005, in part by the Natural Sciences and Engineering Research Council of Canada, and in part by the Fundamental Research Funds for the Central Universities under Grant 531118010509. *(Corresponding author: Bai Li.)*

Bai Li and Yakun Ouyang are with the College of Mechanical and Vehicle Engineering, Hunan University, Changsha 410082, China (e-mail: libai@hnu.edu.cn; yakun@hnu.edu.cn).

Youmin Zhang is with the Department of Mechanical, Industrial and Aerospace Engineering, Concordia University, Montreal QC H3G 1M8, Canada (e-mail: ymzhang@encs.concordia.ca).

Tankut Acarman is with the Computer Engineering Department, Galatasaray University, Istanbul 34349, Turkey (e-mail: tacarman@gsu.edu.tr).

Qi Kong is with the JDX R&D Center, JD.com American Technologies Corporation, Mountain View, CA 94043 USA (e-mail: spiritkong@gmail.com).

Zhijiang Shao is with the College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: szj@zju.edu.cn).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2021.3056346, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3056346

I. Introduction

A. Background and Motivations

SINGLE-VEHICLE trajectory planning refers to finding a trajectory that is kinematically feasible, collision-free from the obstacles, comfortable for the ride of passengers, and optimal in traveling time [1]. An extension of the single-vehicle case is the multi-vehicle trajectory planning (MVTP) task, which additionally requires mutual collision avoidance among the vehicles. Since these additional constraints scale intractably with the vehicle population (commonly known as the curse of dimensionality), a single-vehicle trajectory planner cannot be directly extended to deal with an MVTP problem [2].

This paper aims to find the time-optimal MVTP result for a team of car-like robots when they travel in a tiny indoor environment occupied by static obstacles (Fig. 1). Despite the variety of multi-agent planners, few of them can efficiently handle such a complicated scheme due to the following reasons. First, the existence of obstacles makes the scenario non-convex [3]. Second, the kinematic constraints of each car-like robot are nonlinear. Third, the tiny environment renders high conflicts among the vehicles, i.e., one vehicle may have underlying conflicts with various vehicles. Fourth, the pursuit for a time-optimal solution indicates that the terminal time is not fixed, which augments the difficulties in the concerned MVTP problem. Fifth, each vehicle is never globally assigned a homotopic route *a priori*, and a misleading choice of homotopy class for even one vehicle could make the entire MVTP task fail. These factors, when coupled together, are beyond the capability of the prevalent MVTP planners.

B. Related Works

The existing MVTP methods are classified into *coupled* and *decoupled* categories. The coupled methods generate the vehicles' trajectories all at once, while the decoupled methods divide the original task into subproblems and then combine the derived solutions to form a complete one.

Typical decoupling strategies include prioritization, reaction, and grouping [4]. Prioritization-based planners generate the single-vehicle trajectory in a sequence, and the robots with already planned trajectories are regarded as moving obstacles when planning for each lower-priority vehicle [5], [6]. However, the already planned trajectories leave an insufficient degree of freedom to trajectory planning of remaining robots, and this procedure may cause deadlocks. A reaction-based planner specifies each robot's one-step maneuver ahead in parallel to avoid collisions [7], [8]. Typical reaction-based planners include

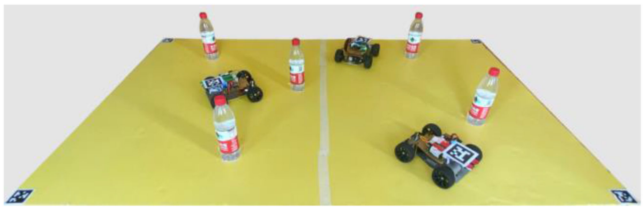

Fig. 1. A 1.8 m $\times$ 1.2 m multi-vehicle trajectory planning setup experimented in this work. Note that the cylinders make the scenario tiny and non-convex.

the Optimal Reciprocal Collision Avoidance (ORCA) method [9], the distributed model predictive control (DMPC) based method [10], and the potential field-based adaptive controllers [11], [12], all of which are promising to enhance the online planning capability. However, the reaction-based methods do not consider the evolution of the neighboring agents' future movements when planning the trajectory for each single robot, thus they are short-sighted in the time horizon and render unnatural travel behaviors [13]. Grouping-based planners cluster the whole team into sub-groups, compute cooperative trajectories for each sub-group independently, and combine the trajectories of all the sub-groups together [14], [15]. However, the sub-groups are not easy to form if all the vehicles are close to each other in a tiny scenario such as in Fig. 1. To conclude, the prevalent three types of decoupled planners can only handle easy MVMP cases wherein the inter-vehicle conflicts are weak.

A coupled MVTP method formulates a single optimal control problem (OCP) gathering the states of all the cooperative vehicles then solves it numerically. Mixed-Integer Linear Program (MILP) [16], Mixed-Integer Quadratic Program (MIQP) [17], and Nonlinear Program (NLP) [4] models were proposed to describe the discretized OCP. As stated in [3], the coupled methods often have well introduced theoretical properties such as guaranteed optimality and being free from the deadlock risks, but intractable volumes of interactions among the vehicles need to be simultaneously handled, especially when the team population is large. Instead of solving the full-scale OCP, facilitation strategies are applied to disperse the underlying difficulties of the full-scale OCP into intermediate problems, which are conquered sequentially. Augugliaro *et al.* [18] adopted the Sequential Convex Programming (SCP) method in multi-UAV motion planning, which linearizes all the nonlinear collision avoidance constraints in the full-scale OCP to form an intermediate Quadratic Program (QP) and successively solves it with updated linearization points until convergence to feasibility is assured. However, since the concerned scenario is non-convex, an intermediate QP easily becomes infeasible because no feasible solution lies in the convexified space (we will further discuss this argument in Section IV with simulation results). If an intermediate problem fails to be solved, the sequential computation process would be blocked. More importantly, the optimum derived by SCP is definitely in homotopy with the initial guess, thus a poor initial guess easily misleads the solution process towards infeasibility. As a remedy for this, Chen *et al.* [3] proposed an incremental Sequential Convex Program (iSCP) method, which adds or updates the linearized collision avoidance constraints one after another. Although iSCP is no longer restricted by the homotopy class issue of the initial guess, yet there is not a recovery strategy if an intermediate QP fails to be solved. Li *et al.* [4] proposed a first-relax-then-recover strategy such that the inter-vehicle collision avoidance constraints are first discarded then added back incrementally in pairs. A similar idea is removing all the non-convex collision avoidance constraints first, then adding them back accumulatively along the timeline [19]. However, in either [4] or [19], the scale of the intermediate problems grows monotonically towards being intractable, and there is not a recovery strategy if an intermediate solution process fails. To summarize, i) coupled MVTP methods have the potential to ensure the solution optimality, but the intractability and non-convexity issues in the coupled problem formulation make the solution process challenging; and ii) the decoupled facilitation strategies are imperfect, especially lacking a failure-recovery operation.

C. Contribution

Describing the concerned MVTP problem as a coupled OCP, this paper introduces a decoupled facilitation strategy named the adaptive-scaling constrained optimization (ASCO) to derive an optimum of the coupled MVTP problem. ASCO divides the intractably scaled constraints in the coupled OCP and conquers them in an iterative framework. Concretely, an intermediate NLP problem is formulated in each iteration, wherein only partial collision avoidance constraints within a certain "range" are imposed; the aforementioned certain range is adjusted adaptively, which promotes the convergence rate, enables recovery from an intermediate NLP failure, and makes the NLP solution process less sensitive to the initial guess.

D. Organization

In the rest of this paper, Section II describes the concerned MVTP scheme as a coupled OCP and provides its discretized form (i.e., an NLP problem) in preparation for numerical solution. Section III introduces the ASCO approach. Simulations and experiments are reported in Section IV. Finally, Section V concludes the paper.

II. Coupled OCP Formulation

A coupled OCP is formulated to describe an MVTP scheme wherein multiple car-like robots travel in a tiny indoor environment involving static circular obstacles. The OCP consists of a cost function and many constraints, typically including the vehicle kinematic constraints, boundary-value constraints, and collision avoidance constraints.

Suppose an MVTP scheme involves $\mathrm{N_V}$ front-steering car-like robots. Since these robots travel at relatively low speeds in an indoor environment, the bicycle model is sufficient to describe their kinematic principle [4]:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x_i(t) \\ y_i(t) \\ v_i(t) \\ a_i(t) \\ \phi_i(t) \\ \theta_i(t) \end{bmatrix} = \begin{bmatrix} v_i(t)\cdot\cos\theta_i(t) \\ v_i(t)\cdot\sin\theta_i(t) \\ a_i(t) \\ jerk_i(t) \\ \omega_i(t) \\ v_i(t)\cdot\tan\phi_i(t)/\mathrm{L_{W}}_i \end{bmatrix},$$

$$i\in\{1,\ldots,\mathrm{N_V}\},\, t\in[0,t_\mathrm{f}]. \tag{1}$$

Herein, t_f denotes the terminal time (unknown *a priori*), represents the mid-point of rear-wheel axle (see point P_i in Fig. 2), $v_i(t)$denotes the velocity of P_i, $a_i(t)$ denotes the acceleration, $jerk_i(t)$ denotes the derivative of acceleration, $\phi_i(t)$ refers to the steering-wheel angle, $\omega_i(t)$ denotes the corresponding angular

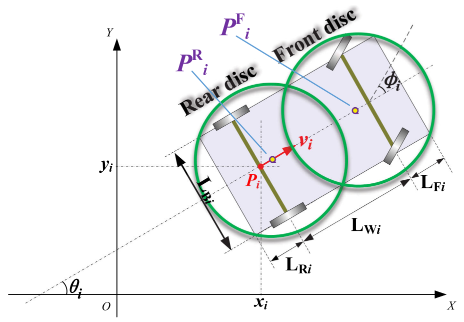

Fig. 2. Kinematics and geometrics of the vehicle *i*. Note that the rectangular shape can be approximated by two discs, whose centers are quartile points along the longitudinal axle.

velocity, $\theta_i(t)$ refers to the orientation angle, and $\mathrm{L}_{\mathrm{W}i}$ denotes the wheelbase. Other geometric parameters, including $\mathrm{L}_{\mathrm{F}i}$ (front overhang length), $\mathrm{L}_{\mathrm{R}i}$ (rear overhang length), and $\mathrm{L}_{\mathrm{B}i}$ (width) are also depicted in Fig. 2.

In addition to (1), boundaries are imposed to describe the mechanical/physical restrictions:

$$\begin{bmatrix} -\mathrm{a}_{\max i} \\ -\mathrm{v}_{\max i} \\ -\mathrm{j}_{\max i} \\ -\Omega_{\max i} \\ -\Phi_{\max i} \end{bmatrix} \leq \begin{bmatrix} a_i(t) \\ v_i(t) \\ jerk_i(t) \\ \omega_i(t) \\ \phi_i(t) \end{bmatrix} \leq \begin{bmatrix} \mathrm{a}_{\max i} \\ \mathrm{v}_{\max i} \\ \mathrm{j}_{\max i} \\ \Omega_{\max i} \\ \Phi_{\max i} \end{bmatrix},$$
$$i \in \{1, \ldots, \mathrm{N_V}\}, t \in [0, t_\mathrm{f}]. \tag{2}$$

Boundary-value constraints are deployed to specify the vehicles' initial and terminal configurations. Suppose that the robots begin to move from a full stop at $t = 0$ and will stop steadily at $t = t_\mathrm{f}$, then

$$\begin{bmatrix} v_i(0) \\ \phi_i(0) \\ a_i(0) \\ jerk_i(0) \\ \omega_i(0) \end{bmatrix} = \begin{bmatrix} v_i(t_\mathrm{f}) \\ \phi_i(t_\mathrm{f}) \\ a_i(t_\mathrm{f}) \\ jerk_i(t_\mathrm{f}) \\ \omega_i(t_\mathrm{f}) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$
$$i \in \{1, \ldots, \mathrm{N_V}\}. \tag{3a}$$

The boundary-value locations and orientations are specified via

$$[x_i(0), y_i(0), \theta_i(0), x_i(t_\mathrm{f}), y_i(t_\mathrm{f}), \theta_i(t_\mathrm{f})] = \mathbf{config}_i,$$
$$i \in \{1, \ldots, \mathrm{N_V}\}, \tag{3b}$$

where $\mathbf{config}_i$ represents the specified starting and terminal configurations of the *i*th vehicle.

The collision avoidance constraints are deployed to restrict vehicle-to-vehicle and vehicle-to-obstacle conflicts. For the convenience of formulation, two discs are used to evenly cover each vehicle's rectangular body as illustrated in Fig. 2. For the *i*th vehicle, the centers of the two discs, denoted as $P_i^\mathrm{F} = (x_i^\mathrm{F}, y_i^\mathrm{F})$ and $P_i^\mathrm{R} = (x_i^\mathrm{R}, y_i^\mathrm{R})$, should be two quartile points along the longitudinal axle of the vehicle [19], i.e.,

$$x_i^\mathrm{F}(t) = x_i(t) + \frac{1}{4}(3\mathrm{L}_{\mathrm{W}i} + 3\mathrm{L}_{\mathrm{F}i} - \mathrm{L}_{\mathrm{R}i}) \cdot \cos\theta_i(t),$$
$$y_i^\mathrm{F}(t) = y_i(t) + \frac{1}{4}(3\mathrm{L}_{\mathrm{W}i} + 3\mathrm{L}_{\mathrm{F}i} - \mathrm{L}_{\mathrm{R}i}) \cdot \sin\theta_i(t),$$
$$x_i^\mathrm{R}(t) = x_i(t) + \frac{1}{4}(\mathrm{L}_{\mathrm{W}i} + \mathrm{L}_{\mathrm{F}i} - 3\mathrm{L}_{\mathrm{R}i}) \cdot \cos\theta_i(t),$$
$$y_i^\mathrm{R}(t) = y_i(t) + \frac{1}{4}(\mathrm{L}_{\mathrm{W}i} + \mathrm{L}_{\mathrm{F}i} - 3\mathrm{L}_{\mathrm{R}i}) \cdot \sin\theta_i(t),$$
$$t \in [0, t_\mathrm{f}]. \tag{4a}$$

The radius of either disc, denoted as R_i^D, is determined by

$$\mathrm{R}_i^\mathrm{D} = \frac{1}{2}\sqrt{\left(\frac{\mathrm{L}_{\mathrm{R}i} + \mathrm{L}_{\mathrm{W}i} + \mathrm{L}_{\mathrm{F}i}}{2}\right)^2 + (\mathrm{L}_{\mathrm{B}i})^2}. \tag{4b}$$

Suppose the environment contains $\mathrm{N_{obs}}$ circular obstacles. The center location of obstacle *k* is denoted as $(\mathrm{x}_k^\mathrm{obs}, \mathrm{y}_k^\mathrm{obs})$, and its radius is $\mathrm{r}_k^\mathrm{obs}$. Following the dual-disc approximation, collisions between vehicles *i* and *j* are avoided by requiring that either disc of vehicle *i* does not overlap with either disc of obstacle *j*:

$$\left(\chi - \mathrm{x}_j^\mathrm{obs}\right)^2 + \left(\gamma - \mathrm{y}_j^\mathrm{obs}\right)^2 \geq (\mathrm{R}_i^\mathrm{D} + \mathrm{r}_j^\mathrm{obs})^2,$$
$$\forall (\chi, \gamma) \in \left\{\left(x_i^\mathrm{F}(t), y_i^\mathrm{F}(t)\right), \left(x_i^\mathrm{R}(t), y_i^\mathrm{R}(t)\right)\right\},$$
$$i \in \{1, \ldots, \mathrm{N_V}\}, j \in \{1, \ldots, \mathrm{N_{obs}}\}, t \in [0, t_\mathrm{f}]. \tag{4c}$$

Similarly, collisions between vehicles *i* and *j* are prohibited via requesting that either disc of vehicle *i* does not overlap with either disc of vehicle *j*:

$$(\chi_1 - \chi_2)^2 + (\gamma_1 - \gamma_2)^2 \geq (\mathrm{R}_i^\mathrm{D} + \mathrm{R}_j^\mathrm{D})^2,$$
$$\forall (\chi_1, \gamma_1) \in \left\{\left(x_i^\mathrm{F}(t), y_i^\mathrm{F}(t)\right), \left(x_i^\mathrm{R}(t), y_i^\mathrm{R}(t)\right)\right\},$$
$$\forall (\chi_2, \gamma_2) \in \left\{\left(x_j^\mathrm{F}(t), y_j^\mathrm{F}(t)\right), \left(x_j^\mathrm{R}(t), y_j^\mathrm{R}(t)\right)\right\},$$
$$i, j \in \{1, \ldots, \mathrm{N_V}\}, i \neq j, t \in [0, t_\mathrm{f}]. \tag{4d}$$

Remark 2.1: The collision avoidance constraints (4c) include $2\mathrm{N_V} \cdot \mathrm{N_{obs}}$ types of inequalities at every instant during $[0, t_\mathrm{f}]$, while the constraints given by (4d) include $2\mathrm{N_V}(\mathrm{N_V} - 1)$ types of inequalities. Thus the overall scale of the collision avoidance constraints is $O(\mathrm{N_V}^2)$, which grows intractably with $\mathrm{N_V}$.

The cost function is deployed to reflect the pursuit for travel efficiency and passenger comfort. A time-energy cost function is presented in (5).

$$J = t_\mathrm{f} + \mathrm{w} \cdot \sum_{i=1}^{\mathrm{N_V}} \left(\int_{\tau=0}^{t_\mathrm{f}} \left(a_i^2(\tau) + v_i^2(\tau) \cdot \omega_i^2(\tau)\right) \cdot \mathrm{d}\tau\right). \tag{5}$$

The first term expects that the entire cooperative traveling process terminates early and the second term is related to the passenger comfort as per the ISO 2631-1 standard [20]. $\mathrm{w} > 0$ is a weighting parameter. Setting w relatively small makes the first term dominant.

With the aforementioned elements summarized, a complete OCP is formulated as

$$\begin{aligned} &\text{Minimize } (5), \\ &\text{s.t.} \\ &\quad \text{Kinematic constraints } (1),\ (2); \\ &\quad \text{Boundary} - \text{value constraints } (3); \\ &\quad \text{Collision avoidance constraints } (4). \end{aligned} \tag{6}$$

Remark 2.2: The unknown variables in (6) include t_{f}, $x_i(t)$, $y_i(t), x_i^{\mathrm{F}}(t), y_i^{\mathrm{F}}(t), x_i^{\mathrm{R}}(t), y_i^{\mathrm{R}}(t), v_i(t), a_i(t), \phi_i(t), \theta_i(t), \omega_i(t)$, and $jerk_i(t), i \in \{1, \ldots, \mathrm{N_V}\}, t \in [0, t_{\mathrm{f}}]$.

Specifying t_{f}, $\omega_i(t)$, and $jerk_i(t)$ would determine the rest variables, and thus specify the trajectories for all the vehicles. In this sense, $\omega_i(t)$ and $jerk_i(t)$ are regarded as control variables, and the remaining variables, except t_{f}, are taken as state variables.

Generally speaking, an analytical solution to (6) cannot be found, a numerical solution should be expected instead. As a basic step of numerical optimization, the first-order explicit Runge-Kutta method to discretize the OCP into an NLP problem. The continuous-time variables in (6) are sampled at ($\mathrm{N_{fe}}+1$) evenly discretized time instances $\{t_k = t_{\mathrm{f}}/\mathrm{N_{fe}} \cdot k |\ k = 0, \ldots, \mathrm{N_{fe}}\}$. For convenience, the NLP problem originated from (6) is referred to as $\mathrm{NLP_{full}}$ in the rest of this paper.

III. Adaptive-Scaling Constrained Optimization

A. Motivations

As $\mathrm{NLP_{full}}$ scales intractably with $\mathrm{N_V}$, directly solving it would be difficult. Even if directly solving $\mathrm{NLP_{full}}$ is applicable, a homotopically wrong initial guess easily misleads the optimization process to infeasibility. Note that the predominant MVTP methods seldom care about the homotopy class issue because the scenario is sparse or the kinematics is simple so that each homotopic choice renders a local optimum. As the scenario in our concerned task is tiny and the vehicle kinematics is non-convex, the fate of directly solving $\mathrm{NLP_{full}}$ relies largely on the quality of the initial guess. Herein, an initial guess specifies each robot's homotopic route, i.e., from which side each robot moves around its companions and the obstacles. Since finding a homotopically appropriate initial guess is extremely difficult [21], we build an iterative optimization framework instead of solving $\mathrm{NLP_{full}}$ directly, hoping to make the NLP solution process insensitive to the homotopy property of the initial guess. We elaborate on our approach and thoughts as follows.

First, although each robot may have conflicts with *different* neighboring robots at *different* instants, *not all* of the robots are subject to mutual collision risks at every moment during $[0, t_{\mathrm{f}}]$. This means that $\mathrm{NLP_{full}}$ contains redundant collision avoidance constraints. Each intermediate NLP problem only contains the partial collision avoidance constraints that are risky to each robot at each discretized time instance as per the initial guess. Herein, the initial guess is the optimum derived by solving the preceding intermediate NLP problem during the iteration. Fig. 3 illustrates the concept of risky constraints in a two-vehicle case, wherein both vehicles are temporarily assumed as mass points for simplicity. Given a risky range [$\mathrm{S_{lb}}$, $\mathrm{S_{ub}}$], the minimum distance between vehicle #2 and obstacle O_2 falls in that range at $t = t_{\mathrm{k}}$, thus the related collision avoidance constraints are added in the current intermediate NLP problem at $t = t_{\mathrm{k}}$. Since the minimum distance between vehicle #2 and obstacle O_1 falls out of the range [$\mathrm{S_{lb}}$, $\mathrm{S_{ub}}$], O_1 is not regarded as risky at $t = t_{\mathrm{k}}$ for vehicle #2. Similarly, the collision avoidance constraints between vehicles #1 and #2 have to be included in the current intermediate NLP problem at $t = t_{\mathrm{k}}$. Discarding constraints out of the range of [$\mathrm{S_{lb}}$, $\mathrm{S_{ub}}$] largely reduces the dimensionality of the NLP problem.

Second, if an intermediate NLP fails to be solved (i.e., converges to infeasibility or does not converge), we may shrink the risky range [$\mathrm{S_{lb}}$, $\mathrm{S_{ub}}$] so that fewer collision avoidance constraints will be considered in the next iteration. The rationale behind the range-shrinking strategy is presented as follows. While solving an intermediate NLP problem, a failure occurs because i) the initial guess is homotopically wrong, or ii) the high-dimension constraints are beyond the ability of the NLP solver. To handle the second failure reason, reducing the range length of [$\mathrm{S_{lb}}$,$\mathrm{S_{ub}}$] renders fewer constraints. To handle the first reason, making $\mathrm{S_{lb}}$ larger leaves out more close-range collision avoidance constraints. This creates opportunities for the robots to "jump" between different homotopic classes, thus making the next iteration less sensitive to the currently available initial guess.

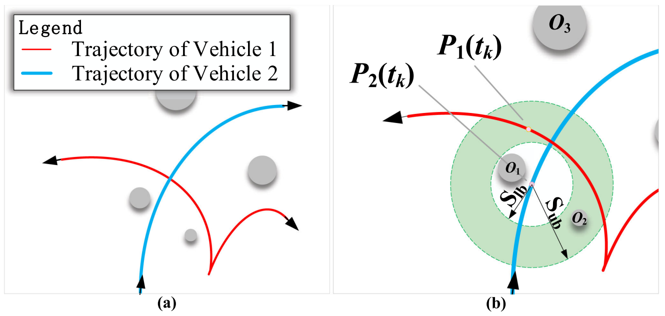

Fig. 3. Rationale of a constraint exclusion strategy in a cooperative planning scheme with two mass-point vehicles: (a) trajectories of the two vehicles in the initial guess; and (b) scale of activated collision avoidance constraints for vehicle #2 at $t = t_{\mathrm{k}}$.According to the initial guess and a specified risky range [$\mathrm{S_{lb}}$, $\mathrm{S_{ub}}$], vehicle #2 is risky at $t = t_{\mathrm{k}}$ with O_2 and vehicle #1, but not O_1. Section III.E formally extends this basic idea to handle car-like robots, wherein each robot is covered by two discs in the shape.

Third, the optimum of an intermediate NLP problem, if derived successfully, may violate some collision avoidance constraints *out of* the currently focused risky range [$\mathrm{S_{lb}}$, $\mathrm{S_{ub}}$]. Under this circumstance, the risky range [$\mathrm{S_{lb}}$, $\mathrm{S_{ub}}$] should be slightly expanded, with the hope of finding a feasible optimum in the next iteration.

The proposal of ASCO is motivated by the aforementioned aspects. It is featured by adaptively updating the scale of the collision avoidance constraints when solving the intermediate NLP problems in an iteration.

B. Overall Architecture of ASCO

The overall principle of ASCO is introduced in Algorithm 1.

In line 3 of Algorithm 1, max_iter denotes the maximum allowable iteration number. In line 4, the collision risks only in the risky range of $[\mathrm{S_{lb}}, \mathrm{S_{ub}}]$ are considered for each vehicle in formulating $\mathrm{NLP_{reduced}}$. The risky range is adjusted adaptively via $\alpha, \beta, \gamma > 0$, which represent the unit steps to change the risky range bounds. The risky range is initialized as [$\mathrm{L_0}$, $\mathrm{L_1}$].

If an intermediate NLP solution failure is encountered, then the lower bound of $[\mathrm{S_{lb}}, \mathrm{S_{ub}}]$ is increased by α, thus making the next-iteration $\mathrm{NLP_{reduced}}$ excludes more close-range collision avoidance constraints so that i) its scale is smaller, and ii) it may escape from the current (wrong) homotopic class.

If successfully derived, the optimum of $\mathrm{NLP_{reduced}}$ (denoted as χ^*) is checked for constraint violation w.r.t $\mathrm{NLP_{full}}$ (line 9). If χ^* does not violate any of the constraints in $\mathrm{NLP_{full}}$, then it is not only the optimum of $\mathrm{NLP_{reduced}}$, but also the optimum of $\mathrm{NLP_{full}}$. The rationale behind this argument is that, if the optimum of a partially constrained problem is feasible to a fully constrained problem, then that optimum is definitely the

Algorithm 1: General structure of ASCO.

```
***Input***: Initial/goal configuration of each vehicle, and
 scenario setup;
***Output***: Cooperative trajectories for the vehicles;
1.  $\chi \leftarrow \mathsf{GenerateInitialGuess}()$;
2.  Initialize $\mathrm{S_{lb}} \leftarrow \mathrm{L_0}, \mathrm{S_{ub}} \leftarrow \mathrm{L_1}$;
3.  **for** $iter = 1 : \mathrm{max\_iter}$,**do**
4.      $\mathrm{NLP_{reduced}} \leftarrow \mathsf{GenerateReducedNLP}(\chi, \mathrm{S_{lb}}, \mathrm{S_{ub}})$;
5.      $[\mathrm{is\_failed}, \chi^*] \leftarrow \mathsf{SolveNLP}(\mathrm{NLP_{reduced}}, \chi)$;
6.      **if** (is_failed), **then**
7.          $\mathrm{S_{lb}} \leftarrow \mathrm{S_{lb}} + \alpha$;
8.      **else**
9.          $\mathrm{is\_feasible} = \mathsf{IsCurrentOptimumFeasible}(\chi^*)$;
10.         **if** (is_feasible), **then**
11.             Output $\chi^*$ and **return** with a success;
12.         **else**
13.             $\chi \leftarrow \chi^*$;
14.             $\mathrm{S_{lb}} \leftarrow \max(\mathrm{S_{lb}} - \beta, \mathrm{L_0})$;
15.             $\mathrm{S_{ub}} \leftarrow \mathrm{S_{ub}} + \gamma$;
16.         **end if**
17.     **end if**
18. **end for**
19. **return** with a failure.
```

optimum of the fully constrained problem as well [19]. That is why any intermediate optimum χ^* that can pass the feasibility check in line 9 is directly output as the final result.

If an intermediate NLP is successfully solved but χ^* does not pass the feasibility check in line 9, then the next-iteration risky range is expanded in both the lower and upper bounds via β and γ (lines 14 and 15) with the goal of passing the feasibility check soon with more constraints included.

Details of the four functions mentioned in Algorithm 1 are introduced in the next four subsections, respectively.

C. Generation of an Initial Guess

An initial guess is generated via $\mathsf{GenerateInitialGuess}()$. Since deriving a homotopically perfect initial guess *a priori* is as difficult as solving the MVTP problem itself, ASCO provides a quick initial guess and counts on the iterative computation stage to get rid of its poor quality. A quick initial guess is derived by combining $\mathrm{N_V}$ single-vehicle trajectories, which means the inter-vehicle collisions are temporarily ignored. The hybrid A* algorithm [22] is adopted to generate each vehicle's path so that it satisfies the vehicle kinematics and ensures to be collision-free from the static obstacles. Thereafter, attaching a time-optimal velocity along each path renders a trajectory. The initial guess of t_f is set to the maximum one among the completion moments of all the trajectories. This means that $(\mathrm{N_V} - 1)$ vehicles may reach their goals earlier than $t = t_\mathrm{f}$ and wait for the last vehicle.

D. Solution Feasibility Check

$\mathsf{IsCurrentOptimumFeasible}(\chi^*)$ checks if χ^* violates any of the constraints in the nominal problem $\mathrm{NLP_{full}}$. Recall that each $\mathrm{NLP_{reduced}}$ has the same kinematic and boundary-value constraints as $\mathrm{NLP_{full}}$, thus an optimum of $\mathrm{NLP_{reduced}}$ satisfies all of the kinematic and boundary-value constraints in $\mathrm{NLP_{full}}$. Therefore, the feasibility checking process only needs to focus on the collision avoidance constraints that $\mathrm{NLP_{full}}$ has but $\mathrm{NLP_{reduced}}$ does not have.

Algorithm 2: $\mathrm{NLP_{reduced}} \leftarrow \mathsf{GenerateReducedNLP}(\chi, \mathrm{S_{lb}}, \mathrm{S_{ub}})$.

```
1.  Initialize $\mathrm{NLP_{reduced}} \leftarrow \mathrm{NLP_{full}}$;
2.  **for** each $t_k \in \{t_\mathrm{f}/\mathrm{N_{fe}} \cdot k | k = 1, \ldots, \mathrm{N_{fe}}\}$, **do**
3.      **for** $i = 1 : \mathrm{N_V}$,**do**
4.          Extract the state vector $[x_i(t_k), y_i(t_k), \theta_i(t_k)]$
            from $\chi$, and denote the corresponding vehicle
            footprint as **FP**$_i$;
5.          **for** $j = (i+1) : \mathrm{N_V}$,**do**
6.              Extract the state vector
                $[x_j(t_k), y_j(t_k), \theta_j(t_k)]$ from $\chi$, and denote
                the corresponding vehicle footprint as **FP**$_j$;
7.              **if** the minimum distance between **FP**$_i$ and
                **FP**$_j$ falls out of [$\mathrm{S_{lb}}$, $\mathrm{S_{ub}}$], **then**
8.                  Remove the vehicle-to-vehicle
                    collision avoidance constraints
                    between vehicles *i* and *j* from
                    $\mathrm{NLP_{reduced}}$ at $t_k$;
9.              **end if**
10.         **end for**
11.         **for** $m = 1 : \mathrm{N_{obs}}$, **do**
12.             Extract the state vector $[\mathrm{x}_m^{\mathrm{obs}}, \mathrm{y}_m^{\mathrm{obs}}, \mathrm{r}_m^{\mathrm{obs}}]$
                from the environment setup, and denote
                the corresponding obstacle footprint as
                **FPobs**$_m$;
13.             **if** the minimum distance between **FP**$_i$ and
                **FPobs**$_m$ falls out of [$\mathrm{S_{lb}}$, $\mathrm{S_{ub}}$], **then**
14.                 Remove the vehicle-to-obstacle
                    collision avoidance constraints
                    between vehicle *i* and obstacle *m* from
                    $\mathrm{NLP_{reduced}}$ at $t_k$;
15.             **end if**
16.         **end for**
17.     **end for**
18. **end for**
19. Output $\mathrm{NLP_{reduced}}$.
```

E. Generation of an Intermediate NLP

$\mathsf{GenerateReducedNLP}(\chi, \mathrm{S_{lb}}, \mathrm{S_{ub}})$ is deployed to generate an intermediate NLP problem (i.e., $\mathrm{NLP_{reduced}}$) based on χ, $\mathrm{S_{lb}}$, and $\mathrm{S_{ub}}$. At each discretized moment t_k, this function seeks for the collision avoidance constraints that are out of the range of $[\mathrm{S_{lb}}, \mathrm{S_{ub}}]$,and excludes them from $\mathrm{NLP_{reduced}}$. The concrete procedures of $\mathsf{GenerateReducedNLP}$are presented in Algorithm 2.

Line 7 of Algorithm 2 requires the calculation of minimum distance between two vehicles. For vehicles *i* and *j* at t_k, the minimum distance is defined by

$$
\begin{aligned}
&arg\ \min \|P - Q\| - \mathrm{R}_i^\mathrm{D} - \mathrm{R}_j^\mathrm{D},\\
&\forall P \in \left\{ \left(x_i^\mathrm{F}(t_k), y_i^\mathrm{F}(t_k)\right),\ \left(x_i^\mathrm{R}(t_k), y_i^\mathrm{R}(t_k)\right) \right\},\\
&\forall Q \in \left\{ \left(x_j^\mathrm{F}(t_k), y_j^\mathrm{F}(t_k)\right),\ \left(x_j^\mathrm{R}(t_k), y_j^\mathrm{R}(t_k)\right) \right\}.
\end{aligned}
\tag{7}
$$

Herein, $\|P - Q\|$ denotes the Euclidean distance between the 2-dim points *P* and *Q*. (7) indicates that the minimum distance is a value no smaller than $-\mathrm{R}_i^\mathrm{D} - \mathrm{R}_j^\mathrm{D}$. The min-distance value in line 13 of Algorithm 2 is calculated similarly.

TABLE I
A LIST OF PARAMETRIC SETTINGS FOR SIMULATIONS

Parameter	Description	Setting
L_{Fi}	Front hang length of vehicle $i \in \{1,...,N_V\}$	0.96 m
L_{Wi}	Wheelbase of vehicle $i \in \{1,...,N_V\}$	2.80 m
L_{Ri}	Rear hang length of vehicle $i \in \{1,...,N_V\}$	0.929 m
L_{Bi}	Width of vehicle $i \in \{1,...,N_V\}$	1.942 m
$a_{\max i}$	Upper bound of $\lvert a_i(t)\rvert$, $i \in \{1,...,N_V\}$	0.5 m/s^2
$v_{\max i}$	Upper bound of $\lvert v_i(t)\rvert$, $i \in \{1,...,N_V\}$	2.5 m/s
$j_{\max i}$	Upper bound of $\lvert jerk_i(t)\rvert$, $i \in \{1,...,N_V\}$	1.0 m/s^3
$\Phi_{\max i}$	Upper bound of $\lvert \phi_i(t)\rvert$, $i \in \{1,...,N_V\}$	0.7 rad
$\Omega_{\max i}$	Upper bound of $\lvert \omega_i(t)\rvert$, $i \in \{1,...,N_V\}$	0.5 rad/s
w	Weight parameter in (5)	10^{-2}
N_{fe}	Number of discretized time instances in forming an NLP via Runge-Kutta method	100
L_0, L_1	Initial values of S_{lb} and S_{ub} in Alg. 1	-4, 2
α, β, γ	Unit step of scale adjustment in Alg. 1	3, 1.3, 0.05
max_iter	Maximum allowable iterations in Alg. 1	100

F. NLP Solution

SolveNLP$(\text{NLP}_{\text{reduced}}, \chi)$ stands for solving $\text{NLP}_{\text{reduced}}$ when warm-started by χ. This work adopts the well-known Interior Point Method (IPM) as the NLP solver. The outputs of SolveNLP include a Boolean status named is_failed and a solution vector χ^*. Herein, χ^* consists of the optimized t_f and the optimized collocation points for the state/control variables.

IV. SIMULATIONS, EXPERIMENTS, AND DISCUSSIONS

A. Simulation Setups

The performance of ASCO is evaluated via simulations, which were executed on an i9-9900 CPU with 32 GB RAM that runs at 3.10 × 2 GHz. An open-source package of IPM, named IPOPT [23] (version 3.13.4, linear solver is set to HSL ma97), is executed in AMPL [24] to solve the NLPs.

A benchmark set containing 100 cases is built. In each case, the vehicles' initial and goal configurations are randomly determined in a $20\text{m} \times 20$ m indoor scenario; the location of each circular obstacle is randomly set; and the radius of each circular obstacle is randomly set in the range of [1 m, 2 m]. In each case, we set $N_V = 10$ and $N_{obs} = 5$. Other common parametric settings are listed in Table I.

B. On the Efficacy of ASCO

The optimization results of the benchmark cases are presented at www.bilibili.com/video/BV19E411E7fU/. The optimized cooperative trajectories for Case #100 are depicted in Fig. 4 as an example. Fig. 5 provides some state profiles in association with the optimized trajectories.

C. Comparing ASCO With Existing MVTP Methods

The efficiency of ASCO is evaluated by comparing it to the state-of-the-art MVTP methods regarding the solution success rate and computational time amongst the 100 benchmark cases. The comparative methods are described in Table II. It deserves

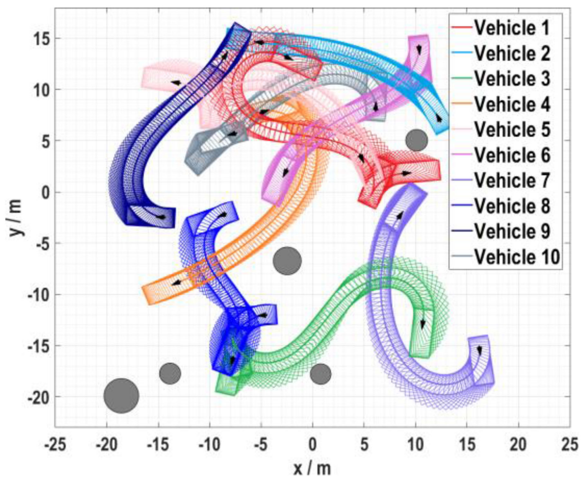

Fig. 4. Optimized cooperative trajectories and footprints of the vehicles in Case #1 ($t_f = 24.116$s).The static obstacles are plotted as grey circles.

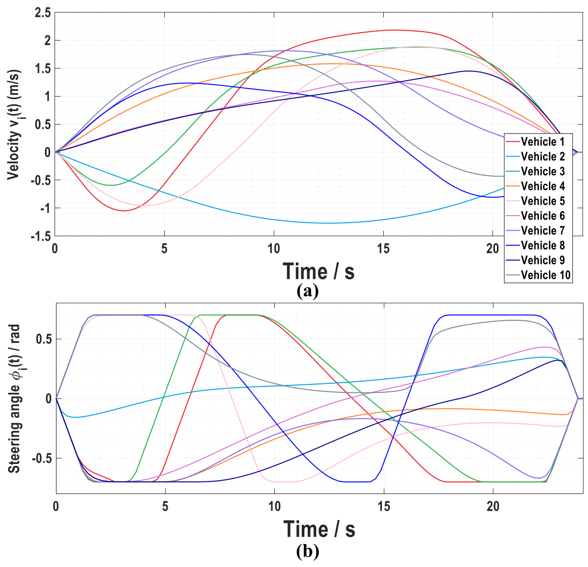

Fig. 5. Optimized velocity and steering angle profiles in Case #1.

to emphasize that some existing methods listed in Table II (such as Algs. 4.1 and 4.2) are nominally developed for robots with linear/linearized kinematic systems, but the kinematic model in this work is non-convex and may not be precisely linearized without a reliable linearization point. Thus we keep the non-convex kinematic constraints as they are in (1) when adopting the existing methods. The comparison results are reported in Table III.

Featured by linearizing all the collision avoidance constraints in each intermediate problem, Algorithm 4.1 is inefficient because the linearized collision avoidance constraints at a poorly guessed linearization point would easily block all the feasibility chances. Interestingly, if we add a white noise (variance = 0.0025) to the optimum derived by ASCO and use it as the Taylor expansion point of SCP, then the success rate of SCP becomes 100%. This indicates that SCP is highly sensitive to the starting point, which is an undesirable property. Compared with SCP, iSCP (Algorithm 4.2) can alter the homotopy class during the iteration, thus it is not limited by the poor quality of straight-line initial guess. However, the success rate is still low

TABLE II
DEFINITION OF COMPARATIVE MVTP METHODS

Algorithm ID	Description
Alg. 4.1	SCP method [18], which linearizes all the collision avoidance constraints via first-order Taylor expansion at the initial guess point to form an intermediate NLP problem, solves it, updates the initial guess, and repeats until a feasible solution to $\mathrm{NLP}_{\mathrm{full}}$ is found.
Alg. 4.2	iSCP method [3], which sequentially adds/updates a linearized collision avoidance constraint in the intermediate NLP problem if that constraint is found to be violated.
Alg. 4.3	A priority-based sequential planning method mentioned in [3], wherein the planned trajectory of a high-priority robot is fixed as a moving obstacle for a low-priority robot. IPM is used to plan each single-vehicle trajectory. The priority is sorted by the Euclidean distance between robots' origin-destination pair in ascending order.
Alg. 4.4	DMPC method [10]. It requires each agent to plan its own trajectory locally, wherein the neighboring agents' behaviors are predicted according to the latest entire initial guess.
Alg. 4.5	A progressively constrained dynamic optimization method [4], which accumulatively adds all the collision avoidance constraints between conflicting pairs of vehicles to the intermediate NLP problems until a feasible solution to $\mathrm{NLP}_{\mathrm{full}}$ is found.
Alg. 4.6	An iterative computation strategy [19], which imposes the collision avoidance constraints along $[0, t_f]$ incrementally to the intermediate NLP problems until a feasible solution to $\mathrm{NLP}_{\mathrm{full}}$ is found.
Alg. 4.7	Solve $\mathrm{NLP}_{\mathrm{full}}$ directly when warm-started by the hybrid A* algorithm as introduced in Section III.C.

TABLE III
COMPARATIVE SIMULATIONS RESULTS

Algorithm ID	Rate of success	Average CPU time / s	Maximum CPU time / s	Standard deviation of CPU time
ASCO	100%	26.02	143.75	22.36
Alg. 4.1	5%	251.02	319.49	83.79
Alg. 4.2	51%	995.53	1452.49	212.71
Alg. 4.3	22%	42.30	112.51	16.08
Alg. 4.4	47%	57.88	206.46	49.56
Alg. 4.5	87%	113.33	1024.75	155.62
Alg. 4.6	71%	1495.89	4186.47	863.04
Alg. 4.7	77%	341.32	1016.86	187.88

because adding/updating the collision avoidance constraints one after another yields a long sequence of intermediate problems and is rather short-sighted. In addition, iSCP lacks a recovery strategy when an intermediate problem shall fail to be solved. The performance of Algorithm 4.3 shows that the priority-based decoupled planners are typically not suitable for the complicated MVTP tasks concerned in this work. If ranking the priorities in descending rather than ascending order, the success rate of Algorithm 4.3 is reached to 35%, which is still imperfect. DMPC (Algorithm 4.4) is inefficient because predicting the neighboring agents' motions accurately is no easier than addressing the originally coupled planning problem alone, thus the distributed computation framework leads to a low convergence rate in dealing with intricate cases. Either Algorithm 4.5 or 4.6 attempts to disperse the difficulties of $\mathrm{NLP}_{\mathrm{full}}$ into pieces and then conquers the pieces incrementally. Like iSCP, neither Algorithm 4.5 nor 4.6 keeps the intermediate NLP problems' scales well-controlled, which monotonically grow towards being intractable. The inefficiency of iSCP, Algorithm 4.5, or 4.6 indicates that dynamically updating the constraints during the iterations is necessary, because a previously added constraint may become redundant if the optimum has largely migrated. Directly solving $\mathrm{NLP}_{\mathrm{full}}$, as Algorithm 4.7 does, is time-consuming and not fully efficient because i) it is difficult to handle all the constraints all at once; and ii) there may not exist a feasible solution that is homotopic with the initial guess.

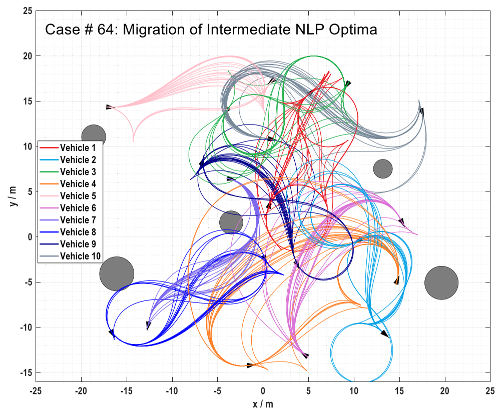

Fig. 6. Migration of optima during the iterations in solving Case #64.

By contrast, ASCO overcomes the aforementioned drawbacks. First, it adaptively updates the set of constraints to keep each intermediate NLP problem tractably scaled, i.e., it does not cumulate the redundant constraints like Algorithm 4.5 or 4.6. Second, it holds a clear strategy to further ease the NLP scale when a solution failure is encountered, which helps to get rid of a poor homotopy class. Third, ASCO enlarges the scale of activated constraints when it expects to be farther sighted to satisfy all the constraints in $\mathrm{NLP}_{\mathrm{full}}$ soon. These are the reasons why ASCO succeeds to find the optima of more benchmark cases with shorter CPU time in comparison with its competitors. To further clarify the second reason, an extra comparison is conducted: we adopt ASCO to solve the 100 cases again with GenerateInitialGuess() replaced by simply connecting the original-destination pairs with lines. As the result, the success rate remains to be 100% while the average CPU time grows to 37.37 seconds.

Before ending this subsection, we use Fig. 6 to illustrate the migration of optima during the 24 iterations in solving Case # 64 via ASCO. The iteration result clearly shows that the intermediate optima can switch to different homotopic classes, thereby making the gradient-based optimization process less reliable with respect to the initial guess.

D. Guidance on Parametric Settings for ASCO

Recall that ASCO is adopted under the nominal parametric settings of $\alpha = 3, \beta = 1.3, \gamma = 0.05, \mathrm{L}_0 = -4$, and $\mathrm{L}_1 = 2$ in the preceding subsection. This subsection provides hints on how these critical parameters would affect the performance of ASCO. Variants of ASCO under different settings are defined in Table IV, wherein each variant only differs from the nominal setting by one parameter.

α denotes the unit step to increase S_{lb}. Setting α larger renders that more close-range constraints are discarded if an intermediate failure occurs. Thus setting α large makes sense in a challenging case, otherwise more iterations (more CPU time) are needed to escape from the current initial guess in a poor homotopic class (see the maximum CPU time derived by the ASCO variant with $\alpha = 1$). β denotes the unit step to decrease S_{lb} when an intermediate NLP is successfully solved. The effect of β is not obvious unless it is set overly small. Setting γ or L_1

TABLE IV
COMPARATIVE SIMULATIONS RESULTS ON ASCO VARIANTS

Algorithm variant feature	Rate of success	Average CPU time / s	Maximum CPU time / s	Standard deviation of CPU time
Nominal ASCO	100%	26.02	143.75	22.36
$\alpha = 1$	100%	27.82	203.94	27.07
$\alpha = 5$	100%	25.95	133.36	20.71
$\beta = 0.7$	100%	26.68	157.61	22.39
$\beta = 2$	100%	25.50	107.17	18.61
$\beta = 5$	100%	25.31	107.01	18.31
$\gamma = 0.1$	100%	28.54	291.45	35.18
$\gamma = 0.5$	100%	59.14	856.78	122.15
$L_1 = 1$	100%	26.42	157.83	26.51
$L_1 = 3$	100%	41.80	339.12	51.44
$L_1 = 5$	100%	117.91	606.04	105.94

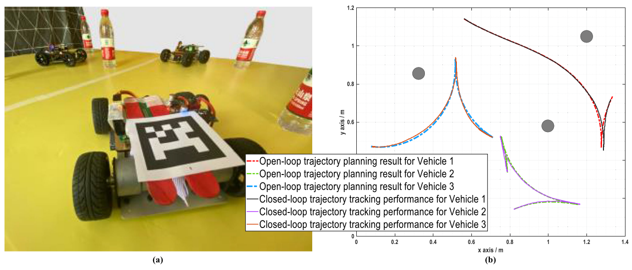

Fig. 7. Experimental platform and results: (a) a swarm of three car-like robots; and (b) closed-loop tracking performance (zoom in to see more clearly).

large yields that more long-range constraints are included in each iteration, thus making the scale of each $\mathrm{NLP}_{\mathrm{reduced}}$ larger and the average CPU time longer.

E. Experimental Setup and Results

Real experiments were conducted with a swarm of three car-like robots in a 1.8 m $\times$ 1.2 m scenario. AprilTags were placed on the top of the robots and the surrounding obstacles for visual localization via a global camera facing the scenario (Fig. 7(a)). The optimized trajectories were sent to the robots through the ZigBee communication technology for closed-loop tracking. Regarding the trajectory tracking in each robot, a PID controller is used for longitudinal tracking, a Pure Pursuit controller is used for lateral tracking, and the control frequency is set to 10 Hz. A typical experimental result is depicted in Fig. 7(b) and more results are presented via the video link provided in Section IV.B.

V. CONCLUSION

This letter presents ASCO as a cooperative trajectory planning method for multiple car-like robots.

A typical application of ASCO is cooperative parking maneuver planning in a parking-lot autonomy system. ASCO may be extended to deal with generic high-dimensional state transfer problems with nonlinear system dynamics, highly conflicting interior states, and non-convex exterior restrictions.

REFERENCES

[1] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016.
[2] B. Li, Y. M. Zhang, Y. Ge, Z. Shao, and P. Li, "Optimal control-based online motion planning for cooperative lane changes of connected and automated vehicles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 3689–3694.
[3] Y. Chen, M. Cutler, and J. P. How, "Decoupled multiagent path planning via incremental sequential convex programming," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 5954–5961.
[4] B. Li, Y. M. Zhang, Z. Shao, and N. Jia, "Simultaneous versus joint computing: A case study of multi-vehicle parking motion planning," *J. Comput. Sci.*, vol. 20, pp. 30–40, 2017.
[5] M. Chen, J. F. Fisac, S. Sastry, and C. J. Tomlin, "Safe sequential path planning of multi-vehicle systems via double-obstacle hamilton-jacobi-isaacs variational inequality," in *Proc. Eur. Control Conf.*, 2015, pp. 3304–3309.
[6] D. R. Robinson, R. T. Mar, K. Estabridis, and G. Hewer, "An efficient algorithm for optimal trajectory generation for heterogeneous multi-agent systems in non-convex environments," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 1215–1222, Apr. 2018.
[7] J. Alonso-Mora, P. Beardsley, and R. Siegwart, "Cooperative collision avoidance for nonholonomic robots," *IEEE Trans. Robot.*, vol. 34, no. 2, pp. 404–420, Apr. 2018.
[8] Y. Zhou, H. Hu, Y. Liu, S. Lin, and Z. Ding, "A real-time and fully distributed approach to motion planning for multirobot systems," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 49, no. 12, pp. 2636–2650, Dec. 2019.
[9] J. Berg, S. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," *Springer Tracts Adv. Robot.*, vol. 70, no. 4, pp. 3–19, 2011.
[10] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 604–611, Apr. 2020.
[11] C. Verginis and D. Dimarogonas, "Adaptive robot navigation with collision avoidance subject to 2nd-order uncertain dynamics," *Automatica*, vol. 123, no. 109303, pp. 1–8, 2021.
[12] D. Dimarogonas, S. Loizou, K. Kyriakopoulos, and M. Zavlanos, "A feedback stabilization and collision avoidance scheme for multiple independent non-point agents," *Automatica*, vol. 42, no. 2, pp. 229–243, 2006.
[13] Y. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non- communicating multiagent collision avoidance with deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 285–292.
[14] H. Xu, S. Feng, Y. Zhang, and L. Li, "A grouping-based cooperative driving strategy for CAVs merging problems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 6125–6136, Jun. 2019.
[15] T. Li *et al.*, "A cooperative lane change model for connected and automated vehicles," *IEEE Access*, vol. 8, pp. 54940–54951, 2020.
[16] T. Schouwenaars, B. De Moor, E. Feron, and J. P. How, "Mixed integer programming for multi-vehicle path planning," in *Proc. Eur. Control Conf.*, 2001, pp. 2603–2608.
[17] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 477–483.
[18] F. Augugliaro, A. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 1917–1922.
[19] B. Li, N. Jia, P. Li, and Y. Li, "Incrementally constrained dynamic optimization: A computational framework for lane change motion planning of connected and automated vehicles," *J. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 557–568, 2019.
[20] B. Sakcak, L. Bascetta, G. Ferretti, and M. Prandini, "An admissible heuristic to improve convergence in kinodynamic planners using motion primitives," *IEEE Control Syst. Lett.*, vol. 4, no. 1, pp. 175–180, Jan. 2020.
[21] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robot. Auton. Syst.*, vol. 88, pp. 142–153, 2017.
[22] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *Int. J. Robot. Res.*, vol. 29, no. 5, pp. 485–501, 2010.
[23] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.
[24] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language For Mathematical Programming*. Brooks/Cole-Thomson Learning, Pacific Grove, 2003.