2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Asynchronous Spatial-Temporal Allocation for Trajectory Planning of Heterogeneous Multi-Agent Systems

Yuda Chen, Haoze Dong, and Zhongkui Li, *Senior Member, IEEE*

***Abstract*— To plan the trajectories of a large-scale heterogeneous swarm, sequentially or synchronously distributed methods usually become intractable due to the lack of global clock synchronization. To this end, we provide a novel asynchronous spatial-temporal allocation method. Specifically, between a pair of agents, the allocation is proposed to determine their corresponding derivable time-stamped space and can be updated in an asynchronous way, by inserting a waiting duration between two consecutive replanning steps. It is theoretically shown that the inter-agent collision is avoided and the allocation ensures timely updates. Comprehensive simulations and comparisons with state-of-the-art baselines validate the effectiveness of the proposed method and illustrate its improvement in completion time and moving distance. Finally, hardware experiments are carried out, where 8 heterogeneous unmanned ground vehicles with onboard computation navigate in cluttered scenarios with high agility.**

Supplementary Materials

Video: https://youtu.be/au3fhqbySOE
Code: https://github.com/CYDXYYJ/ASAP

I. Introduction

Collision-free trajectory planning plays an essential role for a swarm of agents navigating in a shared workspace [1], [2]. The simplest method is to adopt a central coordinator to decide every agent's motion [3], [4]. However, this centralized solution becomes unrealistic and intractable for large-scale swarms, due to the limited interaction range and unbearable computation time. Thus, an increasing number of works, e.g., [5], [6], pursue distributed solutions, where the underlying agents can decide their own actions via local communication with neighbors. Related researches gradually advance from sequentially distributed methods [7] to synchronously concurrent distributed [8], and further to asynchronously distributed ones [9].

For decentralized planning, the commonly-used methods are sequentially distributed ones, e.g., [5], [7], [10], by which the participants replan their trajectories in a sequence. As such, the runtime only linearly increases with the number of agents. By comparison, the concurrent distributed methods, e.g., [11], [2], [12], where different agents concurrently replan trajectories, can further decrease the computation complexity. Concurrent methods can be divided into two categories: synchronous and asynchronous. The former requires both global clock synchronization and timetable coordination to synchronize the calculation duration. This, nevertheless, becomes unrealistic for heterogeneous multi-agent systems, since diversified calculation time and indirect communication are inherent characteristics of those systems.

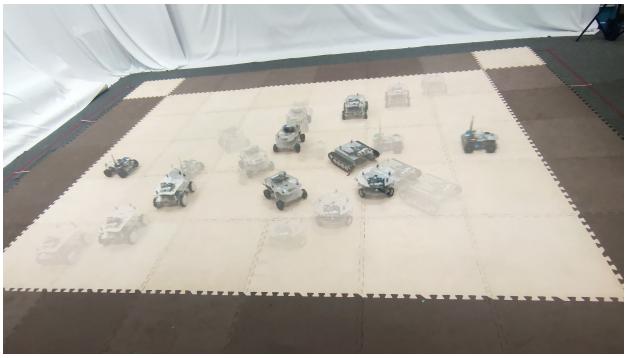

Fig. 1. Eight UGVs are crossing a crowded region.

To fulfill an asynchronous motion planning, the authors in [13] extend the reciprocal velocity obstacles methods [14] to agents with kinodynamic constraints. Nonetheless, the short time horizon in [13] implies that it might not be eligible for high-agility cases. MADER in [6] introduces a novel Check-Recheck deconfliction scheme. However, each agent is only concerned with other agents' current trajectories, neglecting those upcoming ones. Similar issues can be found in [15] and [16]. Meanwhile, [9] ensures safety despite communication delays and interruptions. This method extends the Buffered Voronoi Cells [17] to the asynchronous setting by enlarging the makespan of the constraints which may over-constrain the solution space [15]. Another noteworthy work [18] employs the asynchronous alternating direction method of multipliers (ADMM) to resolve the non-convex constraints. To ensure safety despite packet loss, the method inflates the collision region and continues local ADMM iterations using the latest available predictions of neighbors. However, it lacks a formal proof of collision avoidance. In summary, asynchronously distributed planning methods with rigorous safety guarantees and high efficiency are still in demand.

In this paper a novel method named **A**synchronous **S**patial-**T**emporal **A**llocation (ASTA) is proposed to solve the online trajectory planning problem for heterogeneous multi-agent systems in an asynchronously distributed manner. For each agent, the allocation toward a neighbour is defined as a sequence of time-stamped half space. The allocation is updated asynchronously based on the agents' trajectories through local communication. To achieve this, an interval called

This work was supported in part by the National Science and Technology Major Project under grant 2022ZD0116401; in part by the National Natural Science Foundation of China under grants U2241214, 62373008, T2121002. Corresponding author: Zhongkui Li.

The authors are with the State Key Laboratory for Turbulence and Complex Systems, Department of Mechanics and Engineering Science, College of Engineering, Peking University, Beijing 100871, China.

waiting time is inserted between two consecutive replanning of each agent to allow for allocation updates. To avoid the potential inter-agent collision when the agents adopt different versions of allocations, the update is purposely restricted to a specific duration. It is theoretically proved that by obeying this allocation, the inter-agent collision can be avoided. In addition, the update frequency of the allocation is shown to have a tailored lower bound.

The contributions of the paper are summarized as follows:

- By comparison with other asynchronously distributed methods such as [6] and [18], the planner can predetermine the derivable space and concern other agents' upcoming trajectories. Thus, the planned trajectory theoretically ensures inter-agent collision avoidance without the need of rechecks.
- Compared to [5], [11], the proposed method allows the underlying agents to replan their trajectories by merely considering their own schedules, which makes it applicable for different dynamic models, e.g., double integrators, unicycles, and bicycles.
- Numerical simulations and comparisons with five state-of-the-art methods [5], [6], [8], [11], [2] validate the effectiveness of our method and also illustrate the improvement in the completion time and the moving distances. In the hardware experiments (Fig. 1), 8 heterogeneous unmanned ground vehicles (UGVs) with onboard computation finish two tasks in high agility, where the UGVs' average speed can reach up to 80% of their maximum speed.

II. Problem Formulation

Consider a group of N agents with different kinds of dynamic models in a shared 2D workspace. The objective is to drive these agents from their current states to their corresponding targets x^i_{target}, $i \in \mathcal{N} \triangleq \{1, 2, \ldots, N\}$, without *any* collision under asynchronous local communication.

A. Dynamic Models

For agent $i \in \mathcal{N}$, its state and control input at time t are denoted as $x^i(t) \in \mathbb{R}^m$ and $u^i(t) \in \mathbb{R}^q$, $m, q \in \mathbb{Z}^+$, respectively. Its dynamic model is described as follows:

$$\dot{x}^i(t) = f^i(x^i(t), u^i(t)), \quad x^i(t) \in \mathcal{X}^i,\ u^i(t) \in \mathcal{U}^i, \tag{1}$$

where $\mathcal{X}^i$ and $\mathcal{U}^i$ denote the set of available states and control inputs, respectively; $f^i(\bullet)$ reflects the differential constraint.

Note that the dynamic models of the agents in (1) are heterogeneous, which include the commonly-encountered double integrators, unicycles and bicycles as special cases. Due to space limitations, please refer to the specific form of dynamic models provided in Section II-A of [19].

B. Inter-Agent Collision Avoidance

In order to characterize the inter-agent collision avoidance, we first define the representation of the agents' trajectories.

Definition 1: The trajectory of agent i between time span $[t_a, t_b]$ is defined as

$$\mathcal{T}raj^i(t_a, t_b) \triangleq \left\{ S^i\left(x^i(t)\right) \times t \mid t \in [t_a, t_b] \right\}.$$

The shape of agent i at time t is denoted by a convex polygon $S^i\left(x^i(t)\right)$.

At any time $t > 0$, for any pair of agents i and j, they are collision-free if and only if $S^i\left(x^i(t)\right) \cap S^j\left(x^j(t)\right) = \emptyset$. Accordingly, their trajectories are collision-free if and only if $\mathcal{T}raj^i(0, +\infty) \cap \mathcal{T}raj^j(0, +\infty) = \emptyset$.

C. Problem Statement

In this work, the trajectory planning is carried out in a receding horizon way, wherein $x^i_{n^i}(t)$ is the planned state at t in agent i's n^i-th replanning. Additionally, the planning horizon of the n^i-th replanning is denoted as $T^i_{n^i} \in \mathbb{R}^+$. The time when agent i finishes its n^i-th replanning step is denoted as $t^i_{n^i}$, which is also the starting time of the n^i-th trajectory. The n^i-th replanned trajectory is denoted as $\mathcal{T}raj^i_{n^i}(t^i_{n^i}, +\infty)$. Particularly, for the time beyond the planning horizon, i.e., $t > t^i_{n^i} + T^i_{n^i}$, we enforce $x^i_{n^i}(t) = x^i_{n^i}(t^i_{n^i} + T^i_{n^i})$.

Then, for agent i in its n^i-th replanning, the trajectory generation problem can be formulated as the following optimal control problem (OCP):

$$\begin{aligned}
\min_{x^i_{n^i}(t), u^i_{n^i}(t)} \ & C(x^i_{n^i}, u^i_{n^i}), \\
\text{s.t. } & (1), \\
& x^i(t^i_{n^i}) = \hat{x}^i(t^i_{n^i}), && \text{(2a)} \\
& \mathcal{T}raj^i_{n^i} \cap \mathcal{T}raj^j = \emptyset,\ \forall j \neq i, && \text{(2b)}
\end{aligned}$$

where $t \in [t^i_{n^i}, t^i_{n^i} + T^i_{n^i}]$; $\hat{x}^i(t^i_{n^i})$ is the predicted state at $t^i_{n^i}$; and $C(x^i_{n^i}, u^i_{n^i})$ is the objective function, defined as

$$\begin{aligned}
& C(x^i_{n^i}, u^i_{n^i}) \\
& = \int_{t^i_{n^i}}^{t^i_{n^i} + T^i_{n^i}} \delta x^i_{n^i}(\tau)^{\mathrm{T}} Q\ \delta x^i_{n^i}(\tau) + u^i_{n^i}(\tau)^{\mathrm{T}} P\ u^i_{n^i}(\tau) d\tau,
\end{aligned}$$

which is utilized to drive agent i to its target and penalize the control inputs $u^i_{n^i}(\tau)$, where $\delta x^i_{n^i}(\tau) = x^i_{n^i}(\tau) - x^i_{\text{target}}$, Q and P are the weighted matrices. In the ideal case, the lower-level controller is assumed to perfectly follow the planned trajectory such that $\hat{x}^i(t^i_{n^i}) = \mathcal{T}raj^i_{n^i-1}(t^i_{n^i})$.

Note that the collision avoidance constraints (2b) are impossible to be directly established at the replanning stage, since the exact future motions of other agents are unavailable. To solve this problem, we propose a method to dynamically reformulate constraints (2b) such that the OCP (2) can be formulated and solved in an asynchronous manner while the inter-agent collision is theoretically avoided.

III. Proposed Method

As illustrated in Fig. 2, the proposed method has three main components: i) Spatial-Temporal Allocation (STA), consisting of a sequence of half spaces with time stamps t_i $(i = 1, 2, \ldots)$, is used to determine feasible time-stamped space w.r.t another agent (the yellow part). ii) Between a pair of agents, STA is asynchronously updated based on their trajectories towards a specific duration (the gray part). iii) Trajectory planning via STA aims to confine each agent's planned trajectory in its respective feasible time-stamped space (the red part). These components will be presented in the following subsections.

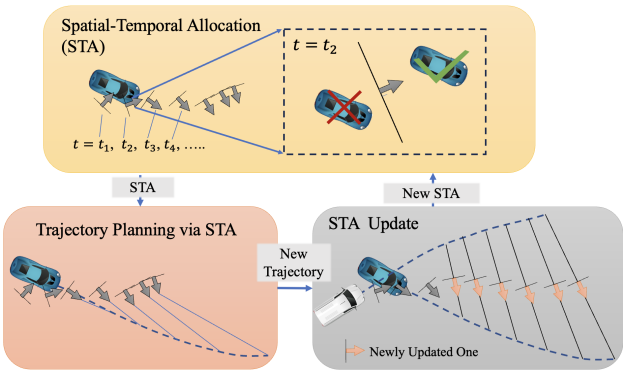

Fig. 2. The components of the proposed method. A solid line and its normal vector are used to demonstrate a half space, where the solid line represents its boundary and the normal vector points toward its safe region.

A. Spatial-Temporal Allocation

STA between agents i and j aims to allocate the time-stamped feasible region for replanning in order to avoid collision between agents. The definition of STA is given below.

Definition 2: The Spatial-Temporal Allocation (STA) of agent i w.r.t. agent j is

$$\mathbb{A}^{ij} = \left\{\mathcal{H}^{ij}(t) \times t \mid t \in [t^{R_1}, +\infty)\right\},$$

where $\mathcal{H}^{ij}(t) \triangleq \{p \mid {a^{ij}}^T(t)\ p > b^{ij}(t)\}$ represents a half space in 2D space, $a^{ij}(t) \in \mathbb{R}^2$ is a normal vector of the boundary of $\mathcal{H}^{ij}(t)$ pointing to the interior of $\mathcal{H}^{ij}(t)$ and $b^{ij}(t) \in \mathbb{R}$ is the offset. We call $\mathcal{H}^{ij}(t)$ a **T**ime-**S**tamped **H**alf **S**pace (TSHS). Moreover, t^{R_1} is the time when these agents establish communication. Conversely, we have $\mathcal{H}^{ji}(t) \triangleq \{p \mid {a^{ji}}^T(t)\ p > b^{ji}(t)\}$, where $a^{ji}(t) = -a^{ij}(t)$, $b^{ji}(t) = -b^{ij}(t)$. As a result, $\mathcal{H}^{ij}(t) \cap \mathcal{H}^{ji}(t) = \emptyset$ and $\mathbb{A}^{ij} \cap \mathbb{A}^{ji} = \emptyset$.

Since STA is updated along with the replanning process, we use $\mathbb{A}^{ij}_m$ to represent the allocation after the $(m-1)$-th update in the sequel.

B. Allocation Update

Allocation update is a core part in our method which keeps the collision avoidance constraints up-to-date. For agent i, there is a waiting time T^i_w between two consecutive trajectory calculation times T^i_c (see Fig. 3). In this work, we particularly enforce $T^i_w > T^i_c$ to ensure a lower bound of the updating frequency.

At time $t^i_{n^i-1}$ when the (n^i-1)-th replanning step is finished, the newly planned trajectory $\mathcal{T}raj^i_{n^i-1}$ is broadcast to other waiting agents. Specifically, if agent j stays at waiting time after its n^j-th replanning, it will reach a *renewal* with agent i based on $\mathcal{T}raj^j_{n^j}$ and $\mathcal{T}raj^i_{n^i-1}$. Fig. 3 provides an illustration. During the waiting time, agent i can also receive other agents' data and reach renewals. A renewal is defined as follows:

$$\mathcal{R}^{ij}_m \triangleq \{\mathcal{H}^{ij}(t) \times t \mid t \in [t^{R_m}_s, +\infty)\},$$

where $t^{R_m}_s \triangleq \max\left\{t^i_{(n^i-1)+1}, t^j_{n^j+1}\right\}$ is the start time of the m-th renewal. Moreover, a part of this renewal from t_a to t_b is denoted as $\mathcal{R}^{ij}_m(t_a, t_b) \triangleq \{\mathcal{H}^{ij}(t) \times t \mid t \in [t_a, t_b)\}$.

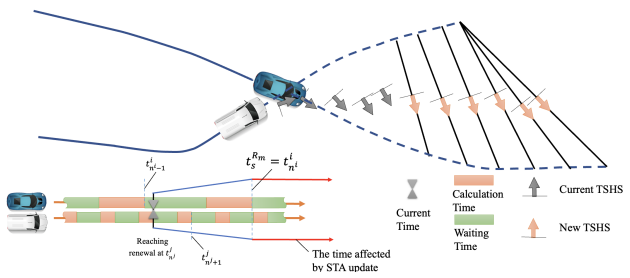

Fig. 3. A demonstration about the STA update for two cars. Both cars stay at waiting time and reach a new renewal, and the TSHS beyond $t^{R_m}_s$ are regenerated to adapt the new situation.

The process that produces a renewal via $\mathcal{T}raj^i_{n^i-1}$ and $\mathcal{T}raj^j_{n^j}$ is to generate hyperplane in order to split the space into two halves $\mathcal{H}^{ij}(t)$ and $\mathcal{H}^{ji}(t)$ based on $S^i(x^i_{n^i-1}(t))$ and $S^j(x^j_{n^j}(t))$ for $t \in [t^{R_m}_s, +\infty)$. According to the separating hyperplane theorem [20], if $S^i(x^i_{n^i-1}(t)) \cap S^j(x^j_{n^j}(t)) = \emptyset$, then we can always find a hyperplane that separates those two polygons. One of the possible implementations of separating hyperplane is using GJK algorithm [21], [11].

Notably, despite the fact that the duration concerned is $[t^{R_m}_s, +\infty)$ for the m-th renewal, only the duration $[t^{R_m}_s, t^{R_m}_e)$, where $t^{R_m}_e \triangleq \max\left\{t^i_{n^i-1} + T^i_{n^i-1}, t^j_{n^j} + T^j_{n^j}\right\}$, needs to be considered. Because for $t > t^{R_m}_e$, $S^i(x^i_{n^i-1}(t)) = S^i(x^i_{n^i-1}(t^{R_m}_e))$ and $S^j(x^j_{n^j}(t)) = S^j(x^j_{n^j}(t^{R_m}_e))$ hold according to the assumption in Section II-C that for $t > t^i_{n^i} + T^i_{n^i}$, $x^i_{n^i}(t) = x^i_{n^i}(t^i_{n^i} + T^i_{n^i})$. Thus, we have $\mathcal{H}^{ij}(t) = \mathcal{H}^{ij}(t^{R_m}_e)$ as well as $\mathcal{H}^{ji}(t) = \mathcal{H}^{ji}(t^{R_m}_e)$ for $t > t^{R_m}_e$.

For STA, when the first renewal starts, it is initialized as $\mathbb{A}^{ij}_1 = \mathcal{R}^{ij}_1(t^{R_1}, +\infty)$. In particular, for the first renewal, we enforce $t^{R_1}_s = t^{R_1}$ to avoid collision during $[t^{R_1}, t^{R_1}_s]$. Then, for a new renewal $\mathcal{R}^{ij}_m$, the STA is updated as follows:

$$\mathbb{A}^{ij}_m = \mathbb{A}^{ij}_{m-1}(t^{R_1}, t^{R_m}_s) \cup \mathcal{R}^{ij}_m(t^{R_m}_s, +\infty), \tag{3}$$

where $\mathbb{A}^{ij}_{m-1}(t^{R_1}, t^{R_m}_s) = \left\{\mathcal{H}^{ij}(t) \times t \mid t \in [t^{R_1}, t^{R_m}_s)\right\}$. In this updating mechanism, STA is only updated for the time beyond $t^{R_m}_s$, which can also be reflected in Fig. 3.

A pair of agents reach their renewal if and only if one of them just finishes its replanning and the other one stays at waiting time. This is a typical asynchronous method and thus called *Asynchronous* Spatial-Temporal Allocation.

C. Trajectory Planning Using STA

In OCP (2), during agent i's n^i-th replanning, given $\mathbb{A}^{ij}_m$ as the latest STA with agent j, the constraints (2b) can be replaced by

$$\mathcal{T}raj^i_{n^i} \subset \mathbb{A}^{ij}_m,\ j \in \mathcal{N}^i, \tag{4}$$

where $\mathcal{N}^i$ is the set of agents that have ever established communication with agent i. Based on this replacement, we

Algorithm 1: The Complete Algorithm

Input: $x^i(0)$, x^i_{target}

1 $\mathcal{T}raj^i_0 = \left\{S\left(x^i(0)\right) \times t \mid t \in [0, +\infty)\right\}$;
2 **while** agent i not reaching target **do**
3 $\quad \mathcal{N}^i \leftarrow$ scan communication network ;
4 $\quad$ send $\mathcal{T}raj^i_{n^i-1}$ to agent $j \in \mathcal{N}^i$;
5 $\quad$ **for** $j \in \mathcal{N}^i$ *concurrently* **do**
6 $\quad\quad$ open receiver in a new thread ;
7 $\quad\quad$ update allocation as (3) if receiving feedback ;
8 $\quad\quad$ close receiver ;
9 $\quad x^i_{n^i} \leftarrow$ Trajectory planning as (5) ;
10 $\quad$ send $x^i_{n^i}$ to the lower-level controller;

further reformulate the OCP (2) as follows:

$$\min_{x^i_{n^i}(t), u^i_{n^i}(t)} \; C(x^i_{n^i}, u^i_{n^i})$$
$$\text{s.t. } (1), (2a),$$
$$S^i(x^i_{n^i}(t)) \in \mathcal{H}^{ij}(t), \tag{5a}$$
$$S^i(x^i_{n^i}(t^i_{n^i} + T^i_{n^i})) \in \mathcal{H}^{ij}(t'), \tag{5b}$$

where $t \in [t^i_{n^i}, t^i_{n^i} + T^i_{n^i}]$ and $t' \in (t^i_{n^i} + T^i_{n^i}, t^{R_m}_e]$.

Note that the constraints (4) consider an intractably infinite time interval $t \in [t^i_{n^i}, +\infty)$. We replace it with constraints (5a) and (5b) which only consider a finite horizon, thus facilitating the application. The reason why OCP (2) with constraints (4) is equal to OCP (5) can be found in [19].

D. The Overall Algorithm

The overall planning algorithm built up by ASTA is outlined in Alg. 1. At the beginning, each agent initializes its trajectory (Line 1), where we enforce that $x^i(0) \in \mathcal{X}^i_e \triangleq \left\{x \mid \exists u, f^i(x, u) = 0\right\}$. Afterwards, the agent scans its communication network to find all neighbors (Line 3) and then broadcasts its trajectory via the communication network (Line 4). Then, for each neighbor, it opens an independent thread to wait for other agents' feedback (Line 6). Once receiving a confirmation signal, the allocation will be updated as in equation (3). Thereafter, the receivers regarding all neighbors are closed (Line 8) and the new trajectory is planned according to OCP (5). However, in this work, the feasibility of the OCP cannot be guaranteed, which highlights an area for our future work. To tackle this, if the given OCP (5) is infeasible, the previous trajectory will be continuously adopted. Finally, the trajectory is sent to the lower-level controller, which controls the agent's motion (Line 10). The whole loop will be carried out indefinitely until this agent reaches its target position.

E. Algorithm Analysis

The proposed method is designed to partially update STA after a specific time $t^{R_m}_s$ as (3). Thanks to such a design, inter-agent collision can be theoretically avoided, which is supported by the following theorem.

Theorem 1: For a pair of agents i and j, if i) agents i's and j's trajectories do not collide with each other when they establish their allocation, and ii) in every following replanning step, they obey their corresponding allocations, then $\forall t > t^{R_1}$, they will not collide with each other.

Besides safety guarantee, another key point is that as the allocation is updated in a triggered way, a lower bound of updating frequency is required. Otherwise, due to the out-of-date allocation, the efficiency of trajectory planning of the swarm will be adversely affected. Therefore, we enforce that the waiting time T^i_w is longer than the computation time T^i_c in Section III-B. Thus, the following property can be obtained.

Theorem 2: Between agents i and j, the lower bound of the frequency of the allocation update is

$$\mathcal{F} = \left(\min\left\{T^i_c, T^j_c\right\} + \max\left\{T^i_c + T^i_w, T^j_c + T^j_w\right\}\right)^{-1}.$$

The proofs of the above two theorems are omitted due to the space limitation. Please refer to [19] for details.

IV. Simulations and Experiments

In this section, we first illustrate the implementation of the proposed method and then validate its performance via numerical simulations and hardware experiments.

A. Algorithm Implementation

This subsection demonstrates how to implement the proposed planning algorithm on digital platforms.

Due to the fact that the OCP (5) is characterized in the continuous-time manner, it requires to be reformulated as a numerical optimization problem via discretization based on a sampling time interval $h^i \in \mathbb{R}$. Accordingly, the length of horizon K^i will be determined as $K^i = \lfloor \frac{T^i}{h^i} \rfloor$. Consequently, in the n^i-th replanning step, the planned state $x^i_{n^i}(t^i_{n^i} + kh^i)$ at time $t^i_{n^i} + kh^i$ and the control input $u^i_{n^i}(t^i_{n^i} + (k-1)h^i)$ at time $t^i_{n^i} + (k-1)h^i$, where $k \in \mathcal{K}^i \triangleq \{1, 2, \ldots, K^i\}$ are chosen as the optimization variables. Towards the constraints (5a) and (5b), they can be rewritten as ${p^i_v}^{\mathrm{T}} a^{ij}(t^i_{n^i} + kh^i) > b^{ij}(t^i_{n^i} + kh^i)$, where $k \in \mathcal{K}^i$, $p^i_v \in V^i_{n^i}(t^i_{n^i} + kh^i)$, $V^i_{n^i}(t)$ is the collection of vertices of polygon regarding the planned state $x^i_{n^i}(t + kh^i)$, $a^{ij}(t^i_{n^i} + kh^i)$ and $b^{ij}(t^i_{n^i} + kh^i)$ represent the hyperplane obtained at time $t^i_{n^i} + kh^i$. Other constraints can be similarly discretized at these time steps.

In terms of implementation, an agent only needs to consider its neighbors within a specified radius. This reduces the number of optimization constraints and achieves manageable computational complexity and admissible scalability. The obstacle avoidance scheme from our previous work [12] can be introduced to apply the proposed method in obstacle environments. Furthermore, in our implementation, an agent can determine the relative time difference between each other via communication. Thus, a global clock synchronization is eliminated. Lastly, we adopt the deadlock resolution scheme in [2] to prevent deadlock.

B. Numerical Simulations

In this subsection, the OCP discretized as in Section IV-A is formulated and resolved by acados [22]. In addition, the proposed communication mechanism is realized via Robot Operating System (ROS). All the simulation cases are run on a single computer with an Intel Core I9@3.5GHz, utilizing multiple programs.

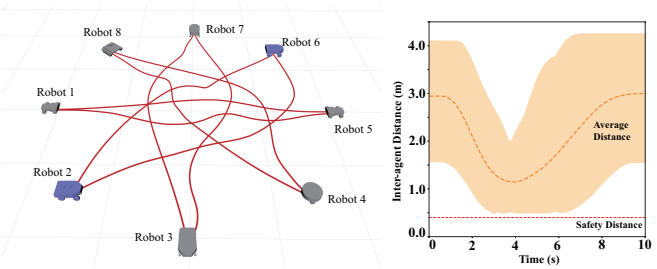

Fig. 4. **Left**: The trajectories of the underlying agents. **Right**: The inter-agent distance.

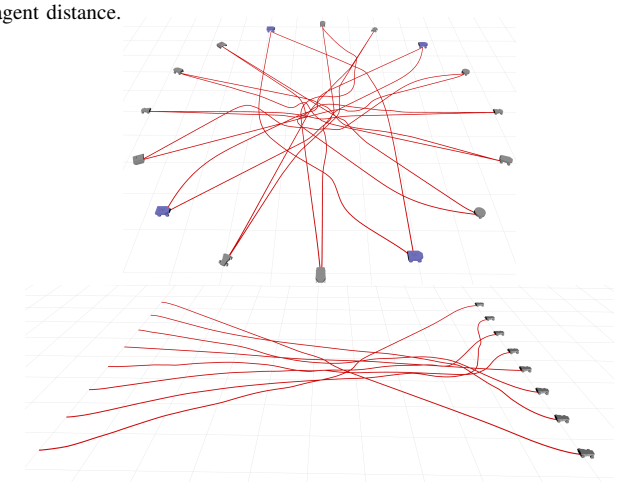

Fig. 5. **Top**: The simulation of 16 agents exchanging their positions. **Bottom**: The simulation of 8 agents changing lanes.

The first simulation example is that 8 agents navigate to their antipodal position in a circle with a diameter of 4.0m. The detailed information about the underlying agents and the simulation results are listed in Table I in [19]. Their diameters are uniformly set as 0.4m, but their maximum speeds range from $0.6\mathrm{m/s}$ to $1.0\mathrm{m/s}$. In this simulation, as elaborated in Section II-A, agents with different dynamic models are considered. The agents' trajectories and inter-agent distance are depicted in Fig. 4, from which it can be observed that the minimum inter-agent distance is around 0.5m, larger than the safety distance 0.4m. It can be seen that safety is guaranteed during the simulation. This task is finished within 8.9s and the maximum transition length is 4.8m as shown in Table I in [19].

Three more complicated scenarios are further simulated. In the first scenario of Fig. 5, we consider a system composed of 16 agents, confirming its effectiveness in handling large-scale systems. In its second scenario, 8 UGVs with bicycle model exchange their lateral positions concurrently, which simulates complicated lane changes in urban road traffic. The result indicates a relatively smooth and fast position exchange, which further exhibits the practicability of our method in real traffic management. We also conducted a simulation to further evaluate the proposed method in an obstacle environment. Interested readers can refer to [19] for more details.

Then, we compare the proposed ASTA with five state-of-the-art methods [5], [2], [8], [11], [6]. The dynamic models of the underlying robots are uniformly determined as double-integrators with maximum velocity and acceleration

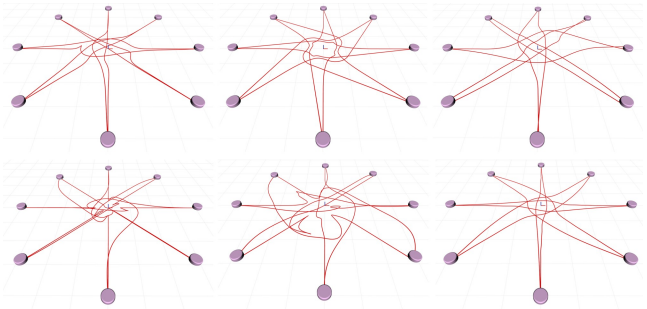

Fig. 6. The results of motion planning. **Top-Left**: Ego-swarm [5]. **Top-Middle**: IMPC-DR [2]. **Top-Right**: DMPC [8]. **Bottom-Left**: LSC [11]. **Bottom-Middle**: MADER [6]. **Bottom-Right**: Proposed method.

TABLE I

COMPARISONS. (Dis: DISTRIBUTION MODE. $T_t[\mathrm{s}]$: MOVING TIME. $T_{\mathrm{COST}}[\mathrm{ms}]$: REPLANNING RUNTIME. $L[\mathrm{m}]$: TRANSITION LENGTH. RESULT REPRESENTATION: MIN/MAX VALUE IN THE SWARM.)

Method	Dis.	T_t	T_{cost}	L
Ego [5]	Seq.	10.0 / 11.6	**3** / **11**	8.33 / 8.80
IMPC-DR [2]	Syn.	9.5 / 10.1	95 / 130	8.57 / 8.79
DMPC [8]	Syn.	9.4 / 10.6	51 / 90	8.43 / 8.89
LSC [11]	Syn.	11.1 / 12.6	15 / 46	9.11 / 9.73
MADER [6]	Asyn.	11.5 / 14.5	24 / 38	9.70 / 11.7
Ours	Asyn.	**9.0** / **9.6**	8 / 21	**8.24** / **8.34**

set as $1.0\mathrm{m/s}$ and $1.5\mathrm{m/s}^2$, respectively. Furthermore, they uniformly have a planning horizon as 2.3s. In this scenario, eight agents need to navigate to their antipodal positions without inter-agent collision. The results are depicted in Fig. 6 and provided in Table I. Executed in a sequential (Seq.) way, the Ego-swarm [5] outperforms other methods in computing time and has a considerably good transition time and distance. The LSC [11], IMPC-DR [2] and DMPC [8] all adopt synchronously (Syn.) concurrent replanning. All of them can complete this task effectively, but with a relatively longer transition distance compared to Ego-Swarm. In addition, replanning in an asynchronous (Asyn.) way, MADER in [6] has a relatively worse performance in this crowded scenario. This may be due to its check-recheck scheme, which results in more conservative trajectories. In contrast, our method provides quicker and shorter transitions in addition to a relatively lower computational cost. In summary, with additional capability to deal with the task in an asynchronous way, the proposed method not only outperforms its asynchronous counterpart in [6], but also performs better than the others in [5], [2], [8], [11].

C. Hardware Experiments

In this subsection, we verify the effectiveness of our method via hardware experiments. In these experiments, 8 UGVs of 5 different hardware models are involved as shown in Fig. 8 in [19]. Their dynamic models include unicycles, double integrators, and kinematic bicycles. The radii of these agents range from 0.15m to 0.26m, and their maximum velocities vary from $0.5\mathrm{m/s}$ to $1.0\mathrm{m/s}$. The architecture of this multi-robot system is depicted in Fig. 8 in [19]. The agents' positions are obtained via the indoor motion capture system OptiTrack and transmitted to other agents

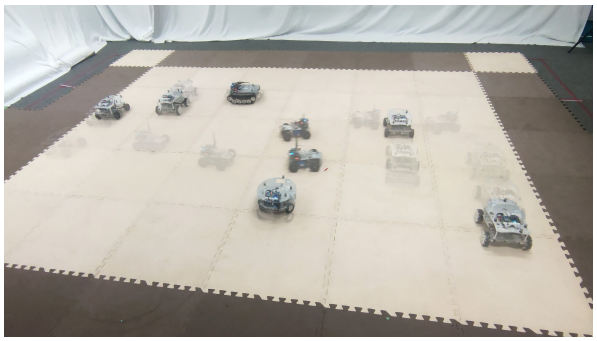

Fig. 7. A snapshot of the experiment of un-signalized intersection.

via WiFi and the ROS-based communication system. The MPC-based trajectory tracker is utilized to track the planned trajectory whose results are sent to the actuator controller. The computation of the trajectory planner, the MPC tracker and the actuator controller is accomplished by a Raspberry Pi 4B onboard computer. To address the communication delays in experiments, we implemented a feedforward compensation mechanism to mitigate their impact.

The first scenario named "crossing" lets eight UGVs cross in a $3.6\mathrm{m} \times 4.2\mathrm{m}$ rectangle ground. Since the maximum radius of the underlying agents reaches up to $0.26\mathrm{m}$, this testing ground is thus crowded. The actual moving snapshot is provided in Fig. 1, where these agents can finish this navigation without any inter-agent collision. Furthermore, the average speed of all UGVs can reach up to 80% of their corresponding maximum velocities, which demonstrates a considerably high agility.

The second scenario is utilized to show the scalability of our method when an agent encounters new neighbors at an un-signalized intersection. Six agents carry on reciprocating motion in a specific area. After tens of seconds, the other two Ackermann UGVs launch their trajectory planning procedures and set up communication with others. Then, these two agents are sent to pass through this area. The result is illustrated in Fig. 7. Two Ackermann UGVs only take less than $5\mathrm{s}$ to finish this task, which means that their average speeds can reach up to almost $0.95\mathrm{m/s}$, i.e., 95% of the maximum velocity. For more information about those experiments, please refer to the supplementary video.

V. Conclusion

This work has proposed a novel distributed asynchronous trajectory planning method called Asynchronous Spatial-Temporal Allocation (ASTA) for multi-agent systems with heterogeneous constrained dynamics. Theoretical analysis, numerical simulations and hardware experiments all validate its effectiveness and performance. Future work will extend the proposed method to address scenarios involving communication delays or malicious neighbors.

References

[1] Z. Liu, M. Guo, W. Bao, and Z. Li, "Fast and adaptive multi-agent planning under collaborative temporal logic tasks via poset products," *Research*, vol. 7, p. Article 0337, 2024.

[2] Y. Chen, M. Guo, and Z. Li, "Deadlock resolution and recursive feasibility in mpc-based multi-robot trajectory generation," *IEEE Transasctions on Automatic Control*, in press, 2024.

[3] W. Hönig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856–869, 2018.

[4] B. Li, Y. Ouyang, Y. Zhang, T. Acarman, Q. Kong, and Z. Shao, "Optimal cooperative maneuver planning for multiple nonholonomic robots in a tiny environment via adaptive-scaling constrained optimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1511–1518, 2021.

[5] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu, and F. Gao, "Swarm of micro flying robots in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm5954, 2022.

[6] J. Tordesillas and J. P. How, "Mader: Trajectory planner in multiagent and dynamic environments," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 463–476, 2021.

[7] A. Richards and J. How, "Decentralized model predictive control of cooperating uavs," in *2004 IEEE Conference on Decision and Control (CDC)*, vol. 4. IEEE, 2004, pp. 4286–4291.

[8] C. E. Luis and A. P. Schoellig, "Trajectory generation for multiagent point-to-point transitions via distributed model predictive control," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 375–382, 2019.

[9] B. Şenbaşlar and G. S. Sukhatme, "Asynchronous real-time decentralized multi-robot trajectory planning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 9972–9979.

[10] D. Morgan, G. P. Subramanian, S.-J. Chung, and F. Y. Hadaegh, "Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming," *International Journal of Robotics Research*, vol. 35, no. 10, pp. 1261–1285, 2016.

[11] J. Park, D. Kim, G. C. Kim, D. Oh, and H. J. Kim, "Online distributed trajectory planning for quadrotor swarm with feasibility guarantee using linear safe corridor," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4869–4876, 2022.

[12] Y. Chen, C. Wang, M. Guo, and Z. Li, "Multi-robot trajectory planning with feasibility guarantee and deadlock resolution: An obstacle-dense environment," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2197–2204, 2023.

[13] J. Alonso-Mora, P. Beardsley, and R. Siegwart, "Cooperative collision avoidance for nonholonomic robots," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 404–420, 2018.

[14] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research: The 14th International Symposium ISRR*. Springer, 2011, pp. 3–19.

[15] K. Kondo, J. Tordesillas, R. Figueroa, J. Rached, J. Merkel, P. C. Lusk, and J. P. How, "Robust mader: Decentralized and asynchronous multiagent trajectory planner robust to communication delay," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1687–1693.

[16] C. Ma, Z. Han, T. Zhang, J. Wang, L. Xu, C. Li, C. Xu, and F. Gao, "Decentralized planning for car-like robotic swarm in cluttered environments," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 9293–9300.

[17] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, online collision avoidance for dynamic vehicles using buffered Voronoi cells," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1047–1054, 2017.

[18] L. Ferranti, L. Lyons, R. R. Negenborn, T. Keviczky, and J. Alonso-Mora, "Distributed nonlinear trajectory optimization for multi-robot motion planning," *IEEE Transactions on Control Systems Technology*, vol. 31, no. 2, pp. 809–824, 2023.

[19] Y. Chen, H. Dong, and Z. Li, "Asynchronous spatial-temporal allocation for trajectory planning of heterogeneous multi-agent systems," *arXiv preprint arXiv:2309.07431*, 2024.

[20] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[21] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[22] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, no. 1, pp. 147–183, 2022.