Distributed Motion Coordination for Multirobot Systems Under LTL Specifications

Pian Yu , *Graduate Student Member, IEEE*, and Dimos V. Dimarogonas , *Senior Member, IEEE*

Abstract—This article investigates the online motion coordination problem for a group of mobile robots moving in a shared workspace, each of which is assigned a linear temporal logic specification. Based on the realistic assumptions that each robot is subject to both state and input constraints and can have only local view and local information, a fully distributed multirobot motion coordination strategy is proposed. For each robot, the motion coordination strategy consists of three layers. An offline layer precomputes the braking area for each region in the workspace, the controlled transition system, and a so-called potential function. An initialization layer outputs an initially safely satisfying trajectory. An online coordination layer resolves conflicts when one occurs. The online coordination layer is further decomposed into three steps. First, a conflict detection algorithm is implemented, which detects conflicts with neighboring robots. Whenever conflicts are detected, a rule is designed to assign dynamically a planning order to each pair of neighboring robots. Finally, a sampling-based algorithm is designed to generate local collision-free trajectories for the robot, which at the same time guarantees the feasibility of the specification. Safety is proven to be guaranteed for all robots at any time. The effectiveness and the computational tractability of the resulting solution is verified numerically by two case studies.

Index Terms—Constraints, distributed control, motion coordination, multirobot systems, safety.

I. INTRODUCTION

ONE challenge for multirobot systems (MRSs) is the design of coordination strategies between robots that enable them to perform operations safely and efficiently in a shared workspace while achieving individual/group motion objectives [1]. This problem was originated from the 80 s and has been extensively investigated since. In recent years, the attention that has been put on this problem has grown significantly due to the emergence of new applications, such as smart transportation and service robotics. The existing literature can be divided into two categories: *path coordination* and *motion coordination*. The former category plans and coordinates the entire paths of all the robots in advance (offline), while the latter category focuses on decentralized and online approaches that allow robots to resolve conflicts online when one occurs[1] [2]. This article aims at developing a fully distributed strategy for multirobot motion coordination (MRMC) with safety guarantees.

Depending on how the controller is synthesized for each robot, the literature concerning MRMC can be further classified into two types: the reactive approach and the planner-based approach. Typical methods that generate reactive controllers consist of the potential-field approach [3], [4], sliding mode control [5], [6], and control barrier functions [7], [8]. These reactive-style methods are fast and operate well in real-time. However, it is well-known that these methods are sensitive to deadlocks that are caused by local minima. Moreover, although these reactive-style methods work well in relatively unconstrained situations, guidance for setting control parameters is not analyzed formally when explicit constraints on the system states and/or inputs are presented [2]. Apart from the above, other reactive methods include the generalized roundabout policy [9] and a family of biologically inspired methods [10].

An early example of the planner-based method is the work of Azarm and Schmidt [11], where a framework for online coordination of multiple mobile robots was proposed. In this framework, MRMC was solved as a sequential trajectory planning problem, where priorities are assigned to robots when conflicts are detected, and then a motion planning algorithm is implemented to generate conflict-free paths. Based on this framework, various applications and different motion planning algorithms are investigated. Guo and Parker [12] proposed a MRMC strategy based on the D* algorithm. In this article, each robot has an independent goal position to reach and know all path information. In [13], a distributed bidding algorithm was designed to coordinate the movement of multiple robots, which focuses on area exploration. In the work of Liu [14], conflict resolution at intersections was considered for connected autonomous vehicles, where each vehicle is required to move along a pre-planned path. A literature review on MRMC can be found in [1].

No matter, which type of controllers is implemented, safety has always been a crucial issue for MRMC. In [7], MRSs with double-integrator dynamics were studied, control barrier functions were proposed to solve the motion coordination problem and safety guarantees were established. However, the velocity

---

[1]In some literature these two terms are used interchangeably. In this article, we try to distinguish between the two as explained previously.

constraints are not dealt with. In [14], safety was stated by assuming that the deceleration that each robot can take is unbounded, yet this assumption may not be realistic for practical applications. In addition, most of the abovementioned literature concerning MRMC considers relatively simple tasks for each robot (e.g., an arrival task from initial state to goal state). However, as robots become more capable, a recent trend in the area of robot motion planning is to assign robots more complex, high-level tasks, such as temporal logic specifications.

In the last few years, multirobot control under linear temporal logic (LTL) specifications has been investigated in [15]–[20]. Most of them consider that the MRS is subject to a global LTL specification, and then an offline motion planning problem is solved in a centralized manner [15]–[18]. In [19], unknown moving obstacles were taken into account. Therefore, online coordination with the moving obstacles was further required. In this article, safety is shown under the assumptions that each robot is moving at a constant velocity and that the local motion planning is feasible. In [20], multirobot plan reconfiguration under local LTL specifications was investigated. A potential-field-based navigation controller was implemented for each robot to guarantee safety. However, the approach is not applicable when state and input constraints are considered.

Motivated by the previous observations, this article investigates the MRMC problem for a group of mobile robots moving in a shared workspace, each of which is assigned a LTL specification. Robots are assumed to have limited sensing capabilities and constraints in both state and input are considered. To cope with these setups, a fully distributed MRMC strategy is proposed. The contributions of this article are summarized as follows.

1) A framework for distributed MRMC under LTL specifications is proposed for each robot, which consists of three layers: an offline precomputation layer, an initialization layer, and an online coordination layer. The online coordination layer is further decomposed into three steps. First, conflicts are detected within the sensing area of each robot. Once conflicts are detected, a rule is applied to assign dynamically a planning order to each pair of neighboring robots. Finally, a sampling-based algorithm is implemented for each robot that generates a local collision-free trajectory, which at the same time satisfies the LTL specification.

2) Safety is established under all circumstances by combining the planner-based controller with an emergency braking controller.

3) As the motion coordination strategy is designed to be fully distributed and each robot considers only local information of neighboring robots, it is totally scalable in the sense that the computational complexity of the strategy does not increase with the number of robots in the workspace.

A comparison between this article and the related literature [15]–[20] is summarized in Table I .

A preliminary version of this article was accepted by the 2020 American Control Conference [21]. Here, we expand this preliminary version in two main directions. First, the framework is generalized to LTL specifications. In the conference version, only reach-avoid type of tasks are considered, and the replanning

TABLE I
COMPARISON OF THIS WORK TO RELATED LITERATURE

Literature	Task[1]	Plan synthesis	Online coordination	Safety[2]
[15]–[18]	global	centralized	no	offline safety
[19]	global	centralized	yes	additional assumptions
[20]	local	distributed	yes	no state and input constraints are considered
This work	local	distributed	yes	no additional assumptions, state and input constraints are considered

[1]Global means that a team LTL specification is assigned to all robots, local means that an individual LTL specification is assigned to each robot.
[2]Safety means robot–robot and robot-obstacle collision avoidance.

problem can be formulated as an optimization problem. However, as the verification of an LTL formula is in general difficult to be conducted online, a local trajectory generation algorithm is designed in this article. Second, safety guarantees are established without the assumption that each robot can take unbounded input in case of emergency. This is done by considering a braking distance at both the conflict detection and coordination steps.

The remainder of this article is organized as follows. In Section II, notation and preliminaries on transition systems, LTL and product automaton are introduced. Section III formalizes the (online) motion coordination problem. Section IV presents the proposed solution in detail, which is verified by two case studies in Section V. Finally, Section VI concludes this article.

II. PRELIMINARIES

A. Notation

Let $\mathbb{R} := (-\infty, \infty)$, $\mathbb{R}_{\geq 0} := [0, \infty)$, $\mathbb{R}_{>0} := (0, \infty)$, and $\mathbb{N} := \{0, 1, 2, \ldots\}$. Denote $\mathbb{R}^n$ as the n-dimensional real vector space, $\mathbb{R}^{n \times m}$ as the $n \times m$ real matrix space. Throughout this article, vectors are denoted in italics $x \in \mathbb{R}^n$, and boldface $\boldsymbol{x}$ is used for continuous-time signals or a sequence of states. Given a continuous-time signal $\boldsymbol{x}$, $x(t)$ denotes the value of $\boldsymbol{x}$ at time t. 0_n denotes a n-dimensional column vector with all elements equal to 0. $[a, b]$ and $[a, b)$ denote closed and right half-open intervals with end points a and b. For $x_1 \in \mathbb{R}^{n_1}, \ldots, x_m \in \mathbb{R}^{n_m}$, the notation $(x_1, x_2, \ldots, x_m) \in \mathbb{R}^{n_1 + n_2 + \cdots + n_m}$ stands for $[x_1^T, x_2^T, \ldots, x_m^T]^T$. Let $|\lambda|$ be the absolute value of a real number λ, $\|x\|$ and $\|A\|$ be the Euclidean norm of vector x and matrix A, respectively. Given a set Ω, 2^Ω denotes its powerset and $|\Omega|$ denotes its cardinality. Given two sets Ω_1, Ω_2, the set $\mathcal{F}(\Omega_1, \Omega_2)$ denotes the set of all functions from Ω_1 to Ω_2. The operators $\cup$ and $\cap$ represent set union and set intersection, respectively. In addition, we use $\wedge$ to denote the logical operator AND and $\vee$ to denote the logical operator OR. The set difference $A \setminus B$ is defined by $A \setminus B := \{x : x \in A \ \wedge \ x \notin B\}$.

Given a vector $x \in \mathbb{R}^n$, define the projection operator $\mathtt{proj}_m(x) : \mathbb{R}^n \to \mathbb{R}^m$ as a mapping from x to its first $m, m \leq n$ components. Given a signal $\boldsymbol{x} : [t_1, t_2] \to \mathbb{R}^n$, define $\mathtt{proj}_m(\boldsymbol{x}) := \{\boldsymbol{x}' : [t_1, t_2] \to$

$\mathbb{R}^m | \boldsymbol{x}'(t) = \text{proj}_m(\boldsymbol{x}(t)), t \in [t_1, t_2]\}$. In addition, we use $\text{dom}(\boldsymbol{x})$ to represent the domain of a signal $\boldsymbol{x}$. Given two signals $\boldsymbol{x}_1 : [t_1, t_2] \to \mathbb{R}^n$ and $\boldsymbol{x}_2 : [t_2, t_3] \to \mathbb{R}^n$, denote by $\boldsymbol{x}_1 \uplus \boldsymbol{x}_2 := \{\boldsymbol{x} : [t_1, t_3] \to \mathbb{R}^n | \boldsymbol{x}(t) = \boldsymbol{x}_i(t), t \in \text{dom}(\boldsymbol{x}_i), i = 1, 2\}$. Given a point $c \in \mathbb{R}^n$, a set $A \subseteq \mathbb{R}^n$ and a constant $r \geq 0$, $\text{dist}(c, A) := \inf_{y \in A}\{\|c - y\|\}$ represents the point-to-set distance and $\mathcal{B}(c, r)$ represents a ball area centered at point c and with radius r. Denote by $\mathcal{B}(A, r) := \cup_{c \in A}\mathcal{B}(c, r)$. Given two sets A, B, define $\text{dist}(A, B) := \inf_{x \in A, y \in B}\{\|x - y\|\}$.

B. Graph Theory

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be a graph with the set of nodes $\mathcal{V} = \{1, 2, \ldots, N\}$, and $\mathcal{E} \subseteq \{(i, j) : i, j \in \mathcal{V}, j \neq i\}$ the set of edges. If $(i, j) \in \mathcal{E}$, then node j is called a neighbor of node i and node j can receive information from node i. The neighboring set of node i is denoted by $\mathcal{N}_i = \{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$. Define $\mathcal{N}_i^+ = \mathcal{N}_i \cup \{i\}$. The graph $\mathcal{G}$ is called undirected if $j \in \mathcal{N}_i \Rightarrow i \in \mathcal{N}_j, \forall j \neq i$. Given an edge $e_k := (i, j) \in \mathcal{E}$, i is called the head of e_k and j is called the tail of e_k. An undirected graph is called connected if for every pair of nodes (i, j), there exists a path which connects i and j, where a path is an ordered list of edges such that the head of each edge is equal to the tail of the following edge.

C. LTL and Büchi Automaton

Let AP be a set of atomic propositions. LTL is based on atomic propositions (state labels $a \in AP$), Boolean connectors like negation $\neg$ and conjunction $\wedge$, and two temporal operators $\bigcirc$ ("next") and U ("until"), and is formed according to the following syntax [22]:

$$\varphi ::= \text{true}|a|\neg\varphi|\varphi_1 \wedge \varphi_2| \bigcirc \varphi|\varphi_1 \mathsf{U}\varphi_2 \tag{1}$$

where $\varphi, \varphi_1, \varphi_2$ are LTL formulas. The Boolean connector disjunction $\vee$, and temporal operators $\Diamond$ ("eventually") and $\Box$ ("always") can be derived as $\varphi_1 \vee \varphi_2 := \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\Diamond\varphi := \text{true}\mathsf{U}\varphi$ and $\Box\phi := \neg\Diamond\neg\phi$. Formal definitions for the LTL semantics and model checking can be found in [22].

Definition 1 (Büchi automaton [23]): A nondeterministic Büchi automaton (NBA) is a tuple $\mathsf{B} = (S, S_0, 2^{AP}, \delta, F)$, where the following conditions hold:
1) S is a finite set of states;
2) $S_0 \subseteq S$ is the set of initial states;
3) 2^{AP} is the input alphabet;
4) $\delta : S \times 2^{AP} \to 2^S$ is the transition function;
5) $F \subseteq S$ is the set of accepting states.

An infinite *run* $\boldsymbol{s}$ of a NBA is an infinite sequence of states $\boldsymbol{s} = s_0 s_1 \ldots$ generated by an infinite sequence of input alphabets $\boldsymbol{\sigma} = \sigma_0 \sigma_1 \ldots \in (2^{AP})^\omega$, where $s_0 \in S_0$ and $s_{k+1} \in \delta(s_k, \sigma_k), \forall k \geq 0$. An infinite run $\boldsymbol{s}$ is called *accepting* if $\text{Inf}(\boldsymbol{s}) \cap F \neq \emptyset$, where $\text{Inf}(\boldsymbol{s})$ is the set of states that appear in $\boldsymbol{s}$ infinitely often. Given a state $s \in S$, define

$$\text{Post}(s) := \{s' \in S : \exists \sigma \in 2^{AP}, s' \in \delta(s, \sigma)\}. \tag{2}$$

Given an LTL formula φ over AP, there is a union of infinite words that satisfy φ, i.e.,

$$\text{Words}(\varphi) = \{\sigma \in (2^{AP})^\omega | \sigma \models \varphi\}$$

where $\models \subseteq (2^{AP})^\omega \times \varphi$ is the satisfaction relation [22].

Lemma 1 ([24]): Any LTL formula φ over AP can be algorithmically translated into a Büchi automaton B_φ over the input alphabet 2^{AP} such that B_φ accepts all and only those infinite runs over AP that satisfy φ.

D. Transition System as Embedding of Continuous-Time Systems

Consider a continuous-time dynamical system

$$\begin{cases} \dot{x} = f(x, u) \\ y = g(x) \end{cases} \tag{3}$$

where $x \in X \subseteq \mathbb{R}^n$ is the state, $u \in U \subseteq \mathbb{R}^m$ is the control, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ describes the dynamics, $y \in Y \subseteq \mathbb{R}^l$ is the output, and $g : \mathbb{R}^n \to \mathbb{R}^l$ is the output function.

Let $\mathcal{U}$ be the set of all measurable functions that take their values in U and are defined on $\mathbb{R}_{\geq 0}$. A curve $\boldsymbol{\xi} : [0, \tau) \to \mathbb{R}^n$ is said to be a trajectory of (3) if there exists an input signal $\boldsymbol{u} \in \mathcal{U}$ satisfying $\dot{\xi}(t) = f(\xi(t), u(t))$ for almost all $t \in [0, \tau)$. A curve $\boldsymbol{\zeta} : [0, \tau) \to \mathbb{R}^l$ is said to be an output trajectory of (3) if $\zeta(t) = g(\xi(t))$ for almost all $t \in [0, \tau)$, where $\boldsymbol{\xi}$ is a trajectory of (3). We use $\xi(\xi_0, \boldsymbol{u}, t)$ and $\zeta(\zeta_0, \boldsymbol{u}, t)$ to denote the trajectory and output trajectory point reached at time t under the input $\boldsymbol{u} \in \mathcal{U}$ from initial condition ξ_0 and ζ_0, respectively. In addition, when $\boldsymbol{u}$ is a constant signal, i.e., $u(t) \equiv \hat{u}, \forall t \in \text{dom}(\boldsymbol{u})$ for a $\hat{u} \in U$, then we define $\xi(\xi_0, \hat{u}, t) := \xi(\xi_0, \boldsymbol{u}, t)$ and $\zeta(\zeta_0, \hat{u}, t) := \zeta(\zeta_0, \boldsymbol{u}, t)$.

The continuous-time system (3) can be represented as an (infinite) transition system $\mathcal{T} = (X, X_0, \Sigma, \to, f, O, g)$, where the following conditions hold:
1) X is the set of states;
2) $X_0 \subseteq X$ is the set of initial states;
3) $\Sigma = \mathcal{U}$ is the set of input functions;
4) $\to : X \times \Sigma \to 2^X$ is the transition relation;
5) $O = Y$ is the set of observations;
6) g is the observation map.

The transition relation $x' \in \to (x, u)$ if and only if $x' = \xi(x, u, \tau)$, where $\tau > 0$ is a given constant. For convenience, $x' \in \to (x, u)$ will be denoted as $x \xrightarrow{u} x'$.

Definition 2 (Controlled transition system): Given a transition system $\mathcal{T} = (X, X_0, \Sigma, \to, f, O, g)$ and a set of atomic propositions AP, we define the controlled transition system (CTS) $\mathcal{T}_c = (X, X_0, AP, \to, L_c)$, where the following condition hold:
1) $L_c : X \to 2^{AP}$ is a labeling function.

The labeling function $L_c(x)$ maps a state x to the finite set of AP, which are true at state x. Given a state $x \in X$, define

$$\text{Post}(x) := \{x' \in X : \exists u \in U, x \xrightarrow{u} x'\}. \tag{4}$$

An infinite *path* of the CTS $\mathcal{T}_c$ is a sequence of states $\boldsymbol{\varrho} = x_0 x_1 x_2 \ldots$ generated by an infinite sequence of inputs $\boldsymbol{u} = u_0 u_1 u_2 \ldots$ such that $x_0 \in X_0$ and $x_k \xrightarrow{u_k} x_{k+1}$ for all $k \geq 0$. Its *trace* is the sequence of atomic propositions that are true in the states along the path, i.e., $\text{Trace}(\boldsymbol{\varrho}) = L_c(x_0)L_c(x_1)L_c(x_2)\ldots$. The satisfaction relation $\boldsymbol{\varrho} \models \varphi$ if and only if $\text{Trace}(\boldsymbol{\varrho}) \in \text{Words}(\varphi)$.

E. Product Automaton and Potential Functions

Definition 3 (Product Büchi automaton [22]): Given a CTS $\mathcal{T}_c = (X, X_0, AP, \rightarrow, L_c)$ and a NBA $\mathsf{B} = (S, S_0, 2^{AP}, \delta, F)$, the product Büchi automaton (PBA) is $\mathcal{P} = \mathcal{T}_c \times \mathsf{B} = (S_p, S_{0,p}, 2^{AP}, \delta_p, F_p)$, where $S_p := X \times S, S_{0,p} := X_0 \times S_0, F_p := (X \times F) \cap S_p$ and the following.

1) $\delta_p \subseteq S_p \times S_p$, defined by $((x, s), (x,' s')) \in \delta_p$ if and only if $x' \in \mathrm{Post}(x)$ and $s' \in \mathrm{Post}(s)$.

Given a state $p \in S_p$, define the projection operator $\mathrm{pj}_X(p) : S_p \to X$ as a mapping from p to its first component $x \in X$. Given a state $x \in X$, define the function $\beta_{\mathcal{P}} : X \to 2^S$, given by

$$\beta_{\mathcal{P}}(x) := \{s \in S : (x, s) \in S_p\} \tag{5}$$

as a mapping from x to the subset of Büchi states S that correspond to x. Denote by $D(p, p')$ the set of all finite runs between state $p \in S_p$ and $p' \in S_p$, i.e.,

$$D(p, p') := \{p_1 p_2 \ldots p_n : p_1 = p, p' = p_n$$
$$(p_k, p_{k+1}) \in \delta_p, \forall k = 1, \ldots, n-1; \forall n \geq 2\}.$$

The state p' is said to be reachable from p if $D(p, p') \neq \emptyset$. The length of a finite run $\boldsymbol{p} = p_1 p_2 \ldots p_n$ in $\mathcal{P}$, denoted by $Lg(\boldsymbol{p})$, is given by

$$Lg(\boldsymbol{p}) := \sum_{i=1}^{n-1} \|\mathrm{pj}_X(p_{i+1}) - \mathrm{pj}_X(p_i)\|.$$

For all $p, p' \in S_p$, the distance between p and p' is defined as follows:

$$d(p, p') = \begin{cases} \min_{\boldsymbol{p} \in D(p,p')} Lg(\boldsymbol{p}), & \text{if } D(p, p') \neq \emptyset \\ \infty, & \text{otherwise.} \end{cases} \tag{6}$$

The following definitions of self-reachable set and potential functions are given in [25].

Definition 4: A set $A \subseteq S_p$ is called *self-reachable* if and only if all states in A can reach a state in A, i.e., $\forall p \in A, \exists p' \in A$ such that $D(p, p') \neq \emptyset$.

Definition 5: For a set $B \subseteq S_p$, a set $C \subseteq B$ is called the *maximal self-reachable set* of B if each self-reachable set $A \subseteq B$ satisfies $A \subseteq C$.

Definition 6 (Potential function of states in $\mathcal{P}$): The potential function of a state $p \in S_p$, denoted by $V_{\mathcal{P}}(p)$ is defined as

$$V_{\mathcal{P}}(p) = \begin{cases} \min_{p' \in F_p^*}\{d(p, p')\}, & \text{if } p \notin F_p^* \\ 0, & \text{otherwise} \end{cases}$$

where F_p^* is the maximal self-reachable set of the set of accepting states F_p in $\mathcal{P}$ and $d(p, p')$ is defined in (6).

Definition 7 (Potential function of states in $\mathcal{T}_c$): Let a state $x \in X$ and a set $M_p \subseteq \beta_{\mathcal{P}}(x)$, where $\beta_{\mathcal{P}}(x)$ is defined in (5). The potential function of x with respect to M_p, denoted by $V_{\mathcal{T}_c}(x, M_p)$ is defined as

$$V_{\mathcal{T}_c}(x, M_p) = \min_{s \in M_p}\{V_{\mathcal{P}}((x, s))\}.$$

Remark 1: If $V_{\mathcal{T}_c}(x, M_p) < \infty$, it means that $\exists s \in M_p$ such that starting from (x, s), there exists a run that reaches a self-reachable accepting state of $\mathcal{P}$.

III. PROBLEM FORMULATION

Consider a group of robots moving in a bounded workspace $\mathbb{W} \subset \mathbb{R}^l$. The dynamics of robot i is given by

$$\dot{x}_i = F_i(x_i, u_i), i \in \mathcal{V} \tag{7}$$

where

$$x_i := (p_i, \zeta_i) \in \mathbb{R}^n$$

represents the state of robot i, which contains its position state $p_i \in \mathbb{R}^l$ and nonposition state $\zeta_i \in \mathbb{R}^{n-l}$ (e.g., orientation and/or velocity), and $u_i \in \mathbb{R}^m$ represents the input of robot i. The function $F_i : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ describes the state evolution of robot i. The output of robot i is the position state, i.e., $y_i = \mathrm{proj}_l(x_i) = p_i, \forall i$.

The state and input of robot i are constrained to the following compact sets:

$$x_i(t) \in \mathbb{X}_i, u_i(t) \in \mathbb{U}_i, \forall t \geq 0. \tag{8}$$

It is assumed that the set $\mathbb{U}_i$ contains the origin for all i.

Denote by $\boldsymbol{\xi}_i : [0, \infty) \to \mathbb{R}^n$ the trajectory of robot i with dynamics given by (7). Then, we further define $\boldsymbol{p}_i : [0, \infty) \to \mathbb{R}^l$ as the position trajectory of robot i, where $p_i(t) = \mathrm{proj}_l(\xi_i(t)), \forall t \in \mathrm{dom}(\boldsymbol{\xi}_i)$. Given a time interval $[t_1, t_2], t_1 < t_2$, the corresponding trajectory and position trajectory are denoted by $\boldsymbol{\xi}_i([t_1, t_2])$ and $\boldsymbol{p}_i([t_1, t_2])$, respectively. Denote by $\boldsymbol{\xi}_i([t, \infty))$ and $\boldsymbol{p}_i([t, \infty))$, the trajectory and the position trajectory of robot i from time t onwards, respectively.

Supposing that the sensing radius of each robot is the same, given by $R > 0$, then the communication graph formed by the group of robots is undirected. The neighboring set of robot i at time t is given by $\mathcal{N}_i(t) = \{j \in \mathcal{V} : \|x_i(t) - x_j(t)\| \leq R, j \neq i\}$, so that $j \in \mathcal{N}_i(t) \Leftrightarrow i \in \mathcal{N}_j(t), \forall i \neq j, \forall t$.

The group of robots are working in a common workspace $\mathbb{W}$, which is populated with a set of closed sets O_i, corresponding to obstacles. Let $\mathbb{O} = \cup_i O_i$, then the free space $\mathbb{F}$ is denoted by $\mathbb{F} := \mathbb{W} \setminus \mathbb{O}$.

Each robot i is subject to its own specification φ_i, which is in the form of a LTL$_{-\mathsf{X}}$ formula that is defined over the workspace $\mathbb{W}$. LTL$_{-\mathsf{X}}$ [26] is a known fragment of LTL, in which the $\bigcirc$ ("next") operator is not allowed. The choice of LTL$_{-\mathsf{X}}$ over LTL is motivated by the fact that LTL [given in (1)] increases expressivity (over LTL$_{-\mathsf{X}}$) only over words with a finite number of repetitions of a symbol, and a word corresponding to a continuous signal will never have a finite number of successive repetitions of a symbol.

Suppose that a cell decomposition is given over the workspace $\mathbb{W}$. The cell decomposition is a partition of $\mathbb{W}$ into finite disjoint convex regions $\Phi := \{X_1, \ldots, X_M\}$ with $\mathbb{W} = \cup_{l=1}^M X_l$. Given a point $p \in \mathbb{W}$, define the map $Q : \mathbb{W} \to \Phi$ as

$$Q(p) := \{X_l \in \Phi : p \in X_l\}$$

which maps a point p into a cell $X_l \in \Phi$ that contains it. Let AP_{φ_i} be the set of atomic propositions specified by φ_i. Define $AP = \cup_{i \in \mathcal{V}} AP_{\varphi_i}$. Then, we have the following assumption.

Assumption 1: The cell decomposition over the workspace $\mathbb{W}$ satisfies

$$L_c(x) = L_c(x'), \forall x, x' : Q(x) = Q(x')$$

where L_c given in Definition 2 is the labeling function.

Assumption 1 means that for all points that are contained in the same cell, the subset of AP that is true at these points is the same. We note that the required cell decomposition can be computed exactly or approximately using many existing approaches ([27, Chs. 4 and 5]).

Given a trajectory $\boldsymbol{\xi}_i$, the notation $\boldsymbol{\xi}_i \models \varphi_i$ means that the trajectory $\boldsymbol{\xi}_i$ satisfies the specification φ_i. Given the position p_i of robot i, we refer to its *footprint* $\phi_i(x_i)$ as the set of points in $\mathbb{W}$ that are occupied by robot i in this position. We note that the footprint of robot i can take into account not only the shape of robot i but also practical issues such as measurement errors.

The objective of this article is to find, for each robot i, a trajectory $\boldsymbol{\xi}_i$ such that $\boldsymbol{\xi}_i \models \varphi_i$ on the premise that safety (no collisions with static obstacles and no interrobot collisions) is guaranteed. Let $t = 0$ be the task activation time of robot $i, \forall i$. Then, the centralized and offline version of the MRMC problem is formulated as follows:

$$\text{find} \quad \{\boldsymbol{\xi}_i([0, \infty))\}_{i \in \mathcal{V}} \tag{9a}$$

subject to

$$(7) \text{ and } (8), \forall i \in \mathcal{V} \tag{9b}$$

$$\boldsymbol{\xi}_i([0, \infty)) \models \varphi_i, \forall i \in \mathcal{V} \tag{9c}$$

$$\phi_i(\boldsymbol{p}_i([0, \infty)) \subset \mathbb{F}, \forall i \in \mathcal{V} \tag{9d}$$

$$\phi_i(p_i(t)) \cap \phi_j(p_j(t)) = \emptyset, \forall i, j \in \mathcal{V}, i \neq j, \forall t. \tag{9e}$$

Constraint (9d) means that the footprint of each robot will not collide with the static obstacles at any time. Constraint (9e) means that the footprint of two different robots can not intersect at any time, thus guaranteeing no interrobot collision occurs. Note that in this article, each robot has only local view and local information, i.e., each robot considers only robots in its neighborhood $\mathcal{N}_i(t)$ at each time t and can have only local information of its neighbors. Therefore, centralized motion coordination can not be conducted. Under these settings, the MRMC problem (9) is broken into local distributed motion coordination problems and solved online for individual robots. Let $\boldsymbol{p}_j([t, t_j^*(t)])$ be the local position trajectory of robot j that is available to robot i at time t, where $t_j^*(t) := \min_{t' > t}\{x_j(t') \notin \mathcal{B}(x_i(t), R)\}$. Then, the (online) motion coordination problem for robot i is formulated as

$$\text{find} \quad \boldsymbol{\xi}_i([t, \infty)) \tag{10a}$$

subject to

$$(7) \text{ and } (8) \tag{10b}$$

$$\boldsymbol{\xi}_i([0, t] \cup [t, \infty)) \models \varphi_i \tag{10c}$$

$$\phi_i(\boldsymbol{p}_i([t, \infty))) \subset \mathbb{F} \tag{10d}$$

$$\phi_i(p_i(t)) \cap \phi_j(p_j(t)) = \emptyset, \forall j \in \mathcal{N}_i(t), \forall t' \in [t, t_j^*(t)] \tag{10e}$$

where $\boldsymbol{\xi}_i([0, t])$ is the history trajectory.

IV. SOLUTION

The proposed solution to the motion coordination problem (10) consists of three layers: 1) an offline pre-computation layer, 2) an initialization layer, and 3) an online coordination layer.
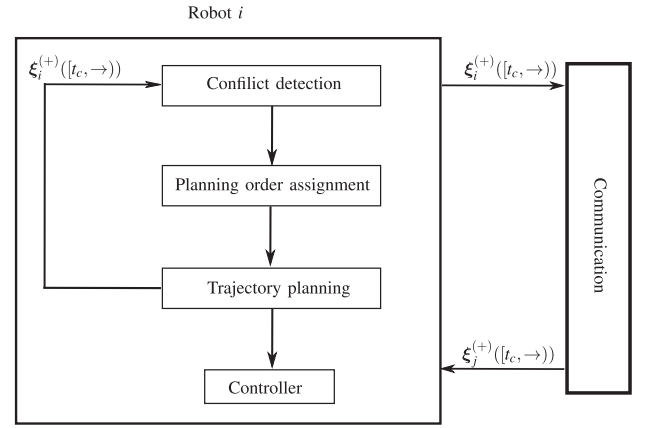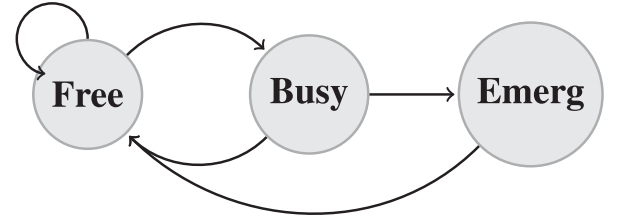

Fig. 1. Structure of robot i.

Fig. 2. Three modes of robot i and the transitions among them.

A. Structure of Each Robot

Before explaining the solution, the structure of each robot is presented (see Fig. 1). Each robot i is equipped with five modules, the conflict detection, the planning order assignment, the trajectory planning, the control, and the communication module. The first four modules work sequentially while the communication module works in parallel with the first four.

During online execution, robot i tries to satisfy its specification safely by resolving conflicts with other robots. This is done by following some mode switching rules encoded into a finite state machine (FSM), see Fig. 2. Each FSM has the following three modes.

1) *Free*: Robot moves as planned. This is the normal mode.
2) *Busy*: Robot enters this mode when conflicts are detected. In this mode, the planning order assignment module and the trajectory planning module are activated.
3) *Emerg*: Robot starts an emergency stop process.

In Fig. 2, the transitions between different modes of the FSM are depicted. Initially, robot i is in *Free* mode. The conflict detection module is activated when the online execution starts. Once conflict neighbors (will be defined later) are detected, robot i enters to *Busy* mode and the planning order assignment and the trajectory planning modules are activated to solve the conflicts, otherwise, robot i stays in *Free* mode. When robot i is in *Busy* mode, it switches back to *Free* mode if the trajectory planning module returns a feasible solution, and the solution will be broadcasted to the robot's neighboring area as well as sent to the controller for execution, otherwise (e.g., no feasible plan is found), robot i switches to *Emerg* mode and a braking controller (defined later) is applied. Note that when robot i switches to *Emerg* mode, it will come to a stop but with power ON. This

means that robot i will continue monitoring the environment and restart (switches back to *Free* mode) when it is possible.

B. Offline Precomputation

1) Braking Controller: As stated in the previous section, when robot i enters *Emerg* mode, it starts an emergency stop process. In our previous work [21], safety in *Emerg* mode is guaranteed by assuming that each robot can take unbounded input when switching to *Emerg* mode. In this article, we consider bounded input in all modes for all robots. Due to this, a braking controller needs to be designed for each robot, and the notions of *braking (position) trajectory* and *braking time and distance* are introduced.

Given the initial state $x_i \in \mathbb{X}_i$ of robot i, define

$$t_i^*(x_i) := \min \quad T \tag{11a}$$

subject to

$$x_i(0) = x_i \tag{11b}$$

$$\dot{x}_i(t) = F_i(x_i(t), u_i(t)) \tag{11c}$$

$$x_i(t) \in \mathbb{X}_i, u_i(t) \in \mathbb{U}_i, t \in [0, T] \tag{11d}$$

$$\dot{p}_i(T) = 0_l \tag{11e}$$

as the minimal time needed to decelerate robot i to zero velocity (i.e., $\dot{p}_i(t) = 0_l$). Let $\boldsymbol{u}_i^*(x_i)$ be the optimal solution of (11). Then, the *braking controller* $\boldsymbol{u}_i^{\mathrm{br}}(x_i)$ is designed as

$$u_i^{\mathrm{br}}(x_i)(t) = \begin{cases} u_i^*(x_i)(t), & \text{if } \dot{p}_i(t) \neq 0_l \\ 0_m, & \text{if } \dot{p}_i(t) = 0_l. \end{cases} \tag{12}$$

Denote by $\boldsymbol{\xi}_i^{\mathrm{br}}(x_i)$ the *braking trajectory* of robot i starting at x_i. Then, we have

$$\xi_i^{\mathrm{br}}(x_i)(t) = \xi_i(x_i, \boldsymbol{u}_i^{\mathrm{br}}(x_i), t), t \in [0, t_i^*(x_i)]. \tag{13}$$

The *braking position trajectory* of robot i, denoted by $\boldsymbol{p}_i^{\mathrm{br}}(x_i)$, is then given by

$$\boldsymbol{p}_i^{\mathrm{br}}(x_i) = \mathrm{proj}_l(\boldsymbol{\xi}_i^{\mathrm{br}}(x_i)). \tag{14}$$

Let $d_i^*(x_i) := \max_{p \in \boldsymbol{p}_i^{\mathrm{br}}(x_i)} \| p - \mathrm{proj}_l(x_i) \|$. Then, we have the following assumption.

Assumption 2: Define the *braking time* T_i^{br} and the *braking distance* D_i^{br} as

$$T_i^{\mathrm{br}} := \max_{x_i \in \mathbb{X}_i} \{ t_i^*(x_i) \} \tag{15}$$

$$D_i^{\mathrm{br}} := \max_{x_i \in \mathbb{X}_i} \{ d_i^*(x_i) \}. \tag{16}$$

Then, we have $0 \le T_i^{\mathrm{br}} < \infty$ and $0 \le D_i^{\mathrm{br}} < \infty, \forall i$.

Remark 2: Assumption 2 means that for any given initial state $x_i \in \mathbb{X}_i$, the maximal time and distance needed for braking with the braking controller (12) are finite. We note that this assumption is not conservative as it is satisfied by many real-life robots. For robots with first-order dynamics (e.g., omni-directional robots, differential drive robots), i.e., the velocity of the robot is controlled, one can derive that $T_i^{\mathrm{br}} = 0$ and $D_i^{\mathrm{br}} = 0$ with the braking controller $\boldsymbol{u}_i^{\mathrm{br}} \equiv 0_m$. For robots with second-order dynamics (e.g., automated vehicles), i.e., the acceleration of the robot is controlled, the braking time T_i^{br} and the braking

distance T_i^{br} are usually determined by the maximum velocity and acceleration.

Example 1: Consider a homogeneous MRS, where the dynamics of each robot i is given by

$$\dot{p}_x^i = v_i \cos \theta_i$$

$$\dot{p}_y^i = v_i \sin \theta_i$$

$$\dot{\theta}_i = \omega_i$$

$$\dot{v}_i = a_i \tag{17}$$

where $p_i := (p_x^i, p_y^i) \in \mathbb{R}^2$ is the position of robot i, $\zeta_i := (\theta_i, v_i)$ represents the nonposition state of robot i, which contains its orientation $\theta_i \in \mathbb{R}$ and velocity $v_i \in \mathbb{R}$. The input u_i is given by $u_i := (\omega_i, a_i)$, where ω_i is the turning rate and a_i is the acceleration.

The velocity and input of robot i are subject to the hard constraints

$$|v_i(t)| \le v_{i,\max}, |\omega_i(t)| \le \omega_{i,\max}, |a_i(t)| \le a_{i,\max} \tag{18}$$

where $v_{i,\max}, \omega_{i,\max}, a_{i,\max} > 0$. Then, the state and input sets are given by

$$\mathbb{X}_i := \{ (p_i, \theta_i, v_i) : p_i \in \mathbb{W}, \theta_i \in \mathbb{R}, |v_i| \le v_{i,\max} \} \tag{19}$$

$$\mathbb{U}_i := \{ (\omega_i, a_i) : |\omega_i| \le \omega_{i,\max}, |a_i| \le a_{i,\max} \}. \tag{20}$$

In this example, we consider two different braking controllers. First, design the braking controller $\boldsymbol{u}_i^{\mathrm{br},\,1}$ as

$$u_i^{\mathrm{br},\,1}(t) = \begin{cases} \left(0, -a_{i,\max} \frac{v_i(t)}{|v_i(t)|} \right), & \text{if } |v_i(t)| \neq 0 \\ 0_2, & \text{if } |v_i(t)| = 0. \end{cases} \tag{21}$$

Then, one can derive the braking time $T_i^{\mathrm{br},\,1} = |v_{i,\max}|/a_{i,\max}$ and the braking distance $D_i^{\mathrm{br},\,1} = |v_{i,\max}|^2/2a_{i,\max}$.

Alternatively, one can design the braking controller $\boldsymbol{u}_i^{\mathrm{br},\,2}$ as

$$u_i^{\mathrm{br},\,2}(t) = \begin{cases} \left((-)\omega_{i,\max}, -a_{i,\max} \frac{v_i(t)}{|v_i(t)|} \right), & \text{if } |v_i(t)| \neq 0 \\ 0_2, & \text{if } |v_i(t)| = 0. \end{cases} \tag{22}$$

Then, one can derive the braking time $T_i^{\mathrm{br},\,2} = |v_{i,\max}|/a_{i,\max}$ and the braking distance

$$D_i^{\mathrm{br},\,2} = \frac{g(v_{i,\max}, \omega_{i,\max}, a_{i,\max})}{\omega_{i,\max}^2}$$

where $g(v_{i,\max}, \omega_{i,\max}, a_{i,\max}) = v_{i,\max}^2 \omega_{i,\max}^2 + 2a_{i,\max}^2 (1 - \cos(v_{i,\max}\omega_{i,\max}/a_{i,\max})) - 2v_{i,\max}\omega_{i,\max}a_{i,\max} \sin(v_{i,\max}\omega_{i,\max}/a_{i,\max})$. Note that using the braking controller (22), one can prove that the minimal braking distance is achieved.

2) Workspace Discretization: Given the set of AP, Assumption 1 guarantees that there exists an observation preserving cell decomposition $\Phi = \{X_1, \dots, X_M\}$ over the workspace $\mathbb{W}$. In general, $X_{l_1}, X_{l_2}, l_1 \neq l_2$ are of different shapes and sizes. For the sake of online coordination, we propose to further discretize each cell $X_l \in \Phi$ into smaller regions. Let $\Xi(X_l) := \{\hat{X}_l^1, \dots, \hat{X}_l^{M_l}\}$ be a partition of X_l into M_l disjoint convex regions. Then, we define $\Xi := \cup_{X_l \in \Phi} \Xi(X_l)$. Ξ can be seen as a finer discretization over the workspace $\mathbb{W}$.

Given a set $S \subseteq \mathbb{W}$, let

$$\hat{Q}(S) := \{ \hat{X} \in \Xi : \hat{X} \cap S \neq \emptyset \} \tag{23}$$

which represents the set of regions in Ξ that intersect with S.

3) Precomputation of Braking Area, CTS, NBA, PBA, and Potential Functions: Given the position p_i of robot i, define

$$\mathbb{S}_i(p_i) := \{x_i \in \mathbb{X}_i : \texttt{proj}_l(x_i) = p_i\}$$

as the subset of states in $\mathbb{X}_i$ that correspond to p_i. In addition, we define the *braking area* of robot i at position p_i as

$$\Phi(p_i) := \cup_{x_i \in \mathbb{S}_i(p_i)} \phi_i(\boldsymbol{p}_i^{\text{br}}(x_i)) \tag{24}$$

where $\phi_i(\boldsymbol{p}_i^{\text{br}}(x_i))$ is the footprint of the braking position trajectory $\boldsymbol{p}_i^{\text{br}}(x_i)$. Let

$$\psi_i(p_i) := \mathcal{B}(\phi_i(p_i), D_i^{\text{br}}). \tag{25}$$

hen, we have the following result.

Proposition 1: The braking area of robot i at position p_i can be overapproximated by the set $\psi_i(p_i)$, i.e.,

$$\Phi(p_i) \subseteq \mathcal{B}(\phi_i(p_i), D_i^{\text{br}}), \forall p_i \in \mathbb{W}.$$

In addition, for each reg ion $\hat{X}_l \in \Xi$, define

$$\psi_i(\hat{X}_l) := \cup_{p_i \in \hat{X}_l} \psi_i(p_i). \tag{26}$$

Each robot i will compute offline a map $M_i : \Xi \to 2^\Xi$, where

$$M_i(\hat{X}_l) := \hat{Q}\left(\psi_i(\hat{X}_l)\right), \forall \hat{X}_l \in \Xi. \tag{27}$$

Intuitively, M_i projects a region $\hat{X}_l$ into a set of regions that might be traversed by the braking position trajectory of robot i if robot i starts an emergency stop process inside $\hat{X}_l$.

Due to the continuity of the dynamics, the state and the input spaces, the transition system that represents (7) is infinite for each robot i. To this end, a probabilistically complete sampling-based algorithm is proposed in [25] to approximate (7) by a finite transition system. Given a sampling interval τ_s, the finite transition system that represents (7) is denoted by $\mathcal{T}_i := (\hat{\mathbb{X}}_i, \hat{\mathbb{X}}_i^0, \mathbb{U}_i, \to_{c,i}, F_i, \mathbb{W}, h)$, where the following conditions hold:

1) $\hat{\mathbb{X}}_i$ collects all sampling points in $\mathbb{X}_i$ that are safe (with respect to the static obstacles), i.e., $\psi_i(\texttt{proj}_l(x_i)) \subseteq \mathbb{F}, \forall x_i \in \hat{\mathbb{X}}_i)$;
2) $\hat{\mathbb{X}}_i^0 \subseteq \hat{\mathbb{X}}_i$;
3) $\to_{c,i} \subseteq \hat{\mathbb{X}}_i \times \mathbb{U}_i \times \hat{\mathbb{X}}_i$.

F_i is given in (7), $\mathbb{W}$ is the workspace as well as the observation space, and $h(\cdot) = \texttt{proj}_l(\cdot)$ is the observation map. Here, $\mathbb{X}_i, \mathbb{U}_i$ are the set of states and inputs, which are defined in (8). The transition relation $(x_i, u_i, x_i') \in \to_{c,i}$ if and only if $x_i' = \xi_i(x_i, u_i, \tau_s)$. Similarly to (4), define

$$\text{Post}(x_i) := \{x_i' \in \hat{\mathbb{X}}_i : \exists u_i \in \mathbb{U}_i, x_i \xrightarrow{u_i} x_i'\}. \tag{28}$$

Once $\mathcal{T}_i$ is obtained, one can further construct the NBA $\mathsf{B}_i := (S_i, S_i^0, 2^{AP_{\varphi_i}}, \delta_i, F_i)$ for the specification φ_i (see Definition 1), the CTS $\mathcal{T}_{c,i} := (\hat{\mathbb{X}}_i, \hat{\mathbb{X}}_i^0, AP_{\varphi_i}, \to_{c,i}, L_{c,i})$ (see Definition 2), and then form the PBA $\mathcal{P}_i = \mathcal{T}_{c,i} \times \mathsf{B}_i$ (see Definition 3). After that, the potential function for $\mathcal{P}_i$ can be computed according to Definition 6.

Remark 3: The cell decomposition Φ of the workspace satisfies $L_c(x) = L_c(x'), \forall Q(x) = Q(x')$. In addition, $AP_{\varphi_i} \subseteq AP, \forall i \in \mathcal{V}$. Therefore, one has $L_{c,i}(x) = L_{c,i}(x'), \forall Q(x) = Q(x'), \forall i \in \mathcal{V}$ and, thus, $\beta_{\mathcal{P}_i}(x) = \beta_{\mathcal{P}_i}(x'), \forall Q(x) = Q(x'), \forall i \in \mathcal{V}$. Then, according to

Definition 6, one can get that if $V_{\mathcal{P}_i}((x, \beta_{\mathcal{P}_i}(x))) < \infty$, then $V_{\mathcal{P}_i}((x,' \beta_{\mathcal{P}_i}(x'))) < \infty, \forall x' \in Q(x)$.

C. Initialization

Before proceeding, the following definition is required.

Definition 8: We call a trajectory $\boldsymbol{\xi}_i$ of (7) *safely satisfy* an LTL formula φ_i if i) $\boldsymbol{\xi}_i \models \varphi_i$ and ii) $\psi_i(p_i) \subseteq \mathbb{F}$, $\forall p_i \in \texttt{proj}_l(\boldsymbol{\xi}_i)$.

At the task activation time $t = 0$, robot i first finds a trajectory $\boldsymbol{\xi}_i^0$ that safely satisfy φ_i. The trajectory planning problem for a single robot can be solved by many existing methods, such as search- or sampling-based [28], [29], automata-based [30], and optimization-based methods [31], [32]. We note that the initial trajectory planning is not the focus of this article. For details about this process, we refer to interested readers to corresponding literatures and the references therein.

The following assumption is needed to guarantee the feasibility of each task specification φ_i.

Assumption 3: Initially, for each robot i, there exists a trajectory $\boldsymbol{\xi}_i^0$ that safely satisfy φ_i.

D. Online Motion Coordination

The initially planned trajectory of each robot does not consider the motion of other robots. Moreover, each robot has only local view and local information (about other robots). Therefore, motion coordination is required during online implementation. Based on the sensing information (about the workspace) and broadcasted information (from neighboring robots), each robot can detect conflicts within its neighborhood and then conduct motion replanning such that conflicts are avoided. In this article, we consider that the conflict detection is conducted at a sequence of sampling instants $\{t_k\}_{k \in \mathbb{N}}$ for each robot i, where $t_{k+1} - t_k \equiv \Delta$.

1) Conflict Detection: Before proceeding, the following notation is introduced. Given a position trajectory $\boldsymbol{p}_i([t_1, t_2])$ and a region $\hat{X}_l \in \Xi$, the function $\Gamma : \mathcal{F}(\mathbb{R}_{\geq 0}, \mathbb{R}^l) \times \Xi \to 2^{\mathbb{R}_{\geq 0}}$, defined as

$$\Gamma(\boldsymbol{p}_i([t_1, t_2]), \hat{X}_l) := \{t \in [t_1, t_2] : p_i(t) \in \hat{X}_l\} \tag{29}$$

gives the time interval that the position trajectory $p_i([t_1, t_2])$ occupies the region $\hat{X}_l$.

Given the position $p_i(t_k)$, $\mathcal{B}(p_i(t_k), R)$ represents the sensing area of robot i at time t_k and $\hat{Q}(\mathcal{B}(p_i(t_k), R))$ represents the set of regions in Ξ that intersect with $\mathcal{B}(p_i(t_k), R)$. Let

$$t_i^{fl}(t_k) := \min_{t > t_k}\{p_i(t) \notin \mathcal{B}(p_i(t_k), R)\}$$

be the first time that robot i leaves its sensing area $\mathcal{B}(p_i(t_k), R)$. Then, define

$$S_i(t_k) := \hat{Q}\left(\boldsymbol{p}_i([t_k, t_i^{fl}(t_k)])\right) \tag{30}$$

as the set of regions traversed by robot i within $\mathcal{B}(p_i(t_k), R)$ until it leaves it at $t_i^{fl}(t_k)$. Moreover, for each $\hat{X}_l \in S_i(t_k)$, the braking area of robot i is contained in $\psi_i(\hat{X}_l)$. Then, we define

$$Res_i(\hat{X}_l) := M_i(\hat{X}_l) \tag{31}$$

as the set of *reserved regions* by robot i in order to safely brake when robot i is within $\hat{X}_l$. According to (29), the time

interval that robot i occupies the region $\hat{X}_l \in S_i(t_k)$ is given by $\Gamma(\boldsymbol{p}_i([t_k, t_i^{fl}(t_k)]), \hat{X}_l)$. In addition, by the continuity of $\boldsymbol{p}_i([t_k, \infty))$, one can conclude that $\Gamma(\boldsymbol{p}_i([t_k, t_i^{fl}(t_k)]), \hat{X}_l)$ is given by one or several disjoint time interval(s) of the form $[a, b), a < b$. Supposing that

$$\Gamma(\boldsymbol{p}_i([t_k, t_i^{fl}(t_k)]), \hat{X}_l) = \cup_{l=1}^m [a_l, b_l) \tag{32}$$

where m is the number of disjoint intervals in $\Gamma(\boldsymbol{p}_i([t_k, t_i^{fl}(t_k)]), \hat{X}_l)$. Denote by $\mathcal{T}_i(\hat{X}_l)$ the time interval that robot i reserves the area $\mathrm{Res}_i(\hat{X}_l)$. Then, it can be overapproximated by

$$\mathcal{T}_i(\hat{X}_l) := \cup_{l=1}^m [a_l, b_l + T_i^{\mathrm{br}}) \tag{33}$$

where T_i^{br} is defined in (15). Then, we have the following definition.

Definition 9: We say that there is a *spatial-temporal conflict* between robot i and j at time t_k if $\exists \hat{X}_l \in S_i(t_k), \hat{X}_{l'} \in S_j(t_k)$ such that $\mathrm{Res}_i(\hat{X}_l) \cap \mathrm{Res}_j(\hat{X}_{l'}) \neq \emptyset$ and $\mathcal{T}_i(\hat{X}_l) \cap \mathcal{T}_j(\hat{X}_{l'}) \neq \emptyset$.

Based on Definition 9, we define the set of conflict neighbors of robot i at time t_k, denoted by $\tilde{\mathcal{N}}_i(t_k)$, as

$$\tilde{\mathcal{N}}_i(t_k) := \{j \in \mathcal{N}_i(t_k) : \exists \hat{X}_l \in S_i(t_k), \hat{X}_{l'} \in S_j(t_k) \text{ s.t.}$$

$$\mathrm{Res}_i(\hat{X}_l) \cap \mathrm{Res}_j(\hat{X}_{l'}) \neq \emptyset \wedge \mathcal{T}_i(\hat{X}_l) \cap \mathcal{T}_j(\hat{X}_{l'}) \neq \emptyset\}. \tag{34}$$

Then, we have the following proposition.

Proposition 2: For robot i, if $\tilde{\mathcal{N}}_i(t_k) = \emptyset$, then one has

i) $\psi_i(p_i(t)) \cap \psi_j(p_j(t)) = \emptyset, \forall j \in \mathcal{N}_i(t_k), \forall t \in [t_k, t_i^{fl}(t_k)]$;

ii) $\phi_i(p_i(t)) \cap \phi_j(p_j(t)) = \emptyset, \forall j \in \mathcal{N}_i(t_k), \forall t \in [t_k, t_i^{fl}(t_k)]$.

Proof: Since T_i^{br} is the maximum time required to decelerate robot i to zero velocity under the braking controller (12), one has that $\Gamma(\boldsymbol{p}_i^{\mathrm{br}}(p_i(t)), \hat{X}_l) \subseteq \mathcal{T}_i(\hat{X}_l), \forall t \in [t_k, t_i^{fl}(t_k)]$. In addition, according to (26), (27), and (31), one has that $\psi_i(p_i(t)) \subseteq \mathrm{Res}_i(\hat{X}_l), \forall p_i(t) \in \hat{X}_l$. That is to say, $\tilde{\mathcal{N}}_i(t_k) = \emptyset$ implies i).

For each region $\hat{X}_l \in \Xi$, define

$$\phi_i(\hat{X}_l) := \cup_{p_i \in \hat{X}_l} \phi_i(p_i).$$

Then, one has $\phi_i(\hat{X}_l) \subseteq \psi_i(\hat{X}_l)$. The time interval that the footprint of robot i occupies $\hat{X}_l$ is given by $\Gamma(\boldsymbol{p}_i([t, t_i^{fl}(t_k)]), \hat{X}_l)$ and $\Gamma(\boldsymbol{p}_i([t, t_i^{fl}(t_k)]), \hat{X}_l) \subseteq \mathcal{T}_i(\hat{X}_l)$. Thus, one can further get that i) implies ii). $\square$

Robot i switches to *Busy* mode if and only if the set of conflict neighbors is nonempty (i.e., $\tilde{\mathcal{N}}_i(t_k) \neq \emptyset$). The conflict detection process is outlined in Algorithm 1.

Remark 4: To implement Algorithm 1, each robot i needs only to broadcast to its neighboring area local information about its plan. To be more specific, which region (e.g., $\hat{X}_l$) within the sensing area $\mathcal{B}(p_i(t_k), R)$ is occupied by robot i and when that happens (i.e., $\Gamma(\boldsymbol{p}_i([t, t_i^{fl}(t_k)]), \hat{X}_l)$). Note that the nonposition information (i.e., $\zeta_i(t)$) of each robot is not required to be broadcasted to the neighbors.

2) Determine Planning Order: Based on the neighboring relation and conflict relation, the graph $\mathcal{G}(t_k) = \{\mathcal{V}, \mathcal{E}(t_k)\}$ formed by the group of robots is naturally divided into one or multiple connected subgraphs, and the motion planning is conducted in parallel within each subgraph in a sequential manner.

Algorithm 1: *conflictDetection.*

Input: $S_j(t_k), \Gamma(\boldsymbol{p}_j([t_k, t_j^{fl}(t_k)]), \hat{X}_l), \forall \hat{X}_l \in S_j(t_k)$ for each $j \in \mathcal{N}_i^+(t_k)$.

Return: Set of conflict neighbors $\tilde{\mathcal{N}}_i(t_k)$.

1: Initialize $\tilde{\mathcal{N}}_i(t_k) = \emptyset$.

2: Compute $\mathrm{Res}_j(\hat{X}_l), \mathcal{T}_j(\hat{X}_l)$ for each $j \in \mathcal{N}_i^+(t_k), \hat{X}_l \in S_j(t_k)$,

3: **for** $j \in \mathcal{N}_i(t_k)$

4:   **if** $\exists \hat{X}_l \in S_i(t_k), \hat{X}_{l'} \in S_j(t_k)$ s.t.
$\mathrm{Res}_i(\hat{X}_l) \cap \mathrm{Res}_j(\hat{X}_{l'}) \neq \emptyset \wedge \mathcal{T}_i(\hat{X}_l) \cap \mathcal{T}_i(\hat{X}_{l'}) \neq \emptyset$

5:     $\tilde{\mathcal{N}}_i(t_k) = \tilde{\mathcal{N}}_i(t_k) \cup \{j\}$,

6:   **end if**

7: **end for**

In order to do that, a planning order needs to be decided for each connected subgraph. In this article, we propose a simple rule to assign priorities between each pair of neighbors.

The number of neighbors and conflict neighbors of robot i at time t_k are given by $|\mathcal{N}_i(t_k)|$ and $|\tilde{\mathcal{N}}_i(t_k)|$, respectively. Then, we have the following definition.

Definition 10: We say that robot i has *advantage* over robot j at time t_k if $\tilde{\mathcal{N}}_j(t_k) \neq \emptyset$ and

1) $|\mathcal{N}_i(t_k)| > |\mathcal{N}_j(t_k)|$; OR

2) $|\mathcal{N}_i(t_k)| = |\mathcal{N}_j(t_k)|$ and $|\tilde{\mathcal{N}}_i(t_k)| > |\tilde{\mathcal{N}}_j(t_k)|$.

Let $\mathcal{Y}_i(t_k)$ be the set of neighbors that have higher priority than robot i at time t_k. The planning order assignment process is outlined in Algorithm 2.

For each neighbor $j \in \mathcal{N}_i(t_k)$, if j is in *Emerg* mode (and, thus, will be viewed as a static obstacle) or $\tilde{\mathcal{N}}_j(t_k) = \emptyset$, then robot j has higher priority (lines 3–5). Otherwise, robot j has higher priority in motion planning if robot j has advantage over robot i (lines 6–8). However, for the special case, i.e., $|\mathcal{N}_i(t_k)| = |\mathcal{N}_j(t_k)|$ and $|\tilde{\mathcal{N}}_i(t_k)| = |\tilde{\mathcal{N}}_j(t_k)|$, neither robot i nor j has advantage over the other. In this case, the priority is determined by the initially uniquely assigned priority score for each robot i (i.e., $P_i^0 \neq P_j^0, \forall i, j$). Denote by P_i^0 the priority score of robot i. We say that robot i has *priority* over j if $P_i^0 > P_j^0$ (lines 9–11).

Proposition 3 (Deadlock-free in planning order assignment): The planning order assignment rule given in Algorithm 2 will result in no cycles, i.e., $\nexists \{q_m\}_1^{\hat{K}}, \hat{K} \geq 2$ such that $q_{\hat{K}} \in \mathcal{Y}_{q_1}(t_k)$ and $q_{m-1} \in \mathcal{Y}_{q_m}(t_k), \forall m = 2, \ldots, \hat{K}$.

Remark 5: The rationale behind the rule can be explained as follows. Since the motion planning is conducted sequentially within each subgraph based on the priority order obtained in Algorithm 2, then the total time required to complete the motion planning is given by $K\mathcal{O}(dt)$, where $\mathcal{O}(dt)$ represents the time complexity of one round of motion planning and K represents the number of rounds (if multiple robots conduct motion planning in parallel, it is counted as one round), which is determined by the priority assignment rule being used (e.g., if fixed priority is used, the number of rounds is $K = N$). In our rule, we assign the robot with more neighbors or more conflict neighbors the higher priority, and in this way, we try to minimize the number of rounds needed.

Algorithm 2: *planningOrderAssignment.*

Input: $\mathcal{N}_j(t_k), \tilde{\mathcal{N}}_j(t_k), P_j^0, j \in \mathcal{N}_i(t_k) \cup \{i\}$.
Return: Set of higher priority neighbors $\mathcal{Y}_i(t_k)$.
1: Initialize $\mathcal{Y}_i(t_k) = \emptyset$.
2: **for** $j \in \mathcal{N}_i(t_k)$ **do,**
3:   **if** j is in **Emerg** mode or $\tilde{\mathcal{N}}_j(t_k) = \emptyset$ **then,**
4:     $\mathcal{Y}_i(t_k) = \mathcal{Y}_i(t_k) \cup j$,
5:   **else**
6:     **if** j has advantage over i **then,**
7:       $\mathcal{Y}_i(t_k) = \mathcal{Y}_i(t_k) \cup j$,
8:     **else**
9:       **if** neither robot i nor j has advantage over the other and $P_j^0 > P_i^0$ **then,**
10:        $\mathcal{Y}_i(t_k) = \mathcal{Y}_i(t_k) \cup j$,
11:      **end if**
12:    **end if**
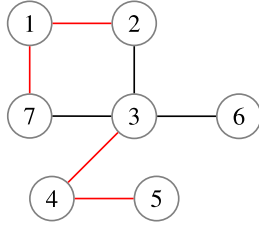13: **end if**
14: **end for**

Fig. 3.   Communication and conflict graph, where both the black and red lines represent communication relation and the red lines represent conflict relation.

Example 2: Consider a group of seven robots, whose communication relation and conflict relation (at time t_k) are depicted in Fig. 3 . According to the proposed planning order assignment rule, the motion planning can be completed in 3 rounds, where robots 1 and 3 are in the first round, robots 2, 4, and 7 are in the second round, and robot 5 is in the third round. Note that no motion planning is required for robot 6 since robot 6 has no conflict neighbor.

E. Motion Planning

Before starting to plan, robot i needs to wait for the updated plan from the set of neighbors that have higher priority than robot i (i.e., $j \in \mathcal{Y}_i(t_k)$) and consider them as moving obstacles. For those neighbors $j \in \mathcal{Y}_i(t_k)$, denote by $\boldsymbol{\xi}_j^+([t_k, \infty))$ (correspondingly $\boldsymbol{p}_j^+([t_k, \infty))$) the updated trajectory (position trajectory) of robot j at time t_k and let $t_j^{fl+}(t_k)$ be the first time that robot j leaves its sensing area $\mathcal{B}(p_j(t_k), R)$ according to the updated position trajectory $\boldsymbol{p}_j^+([t_k, \infty))$. Then, similar to (30), one can define

$$S_j^+(t_k) := \hat{Q}(\boldsymbol{p}_j^+([t_k, t_j^{fl+}]))$$

as the updated set of regions traversed by robot j within $\mathcal{B}(p_j(t_k), R)$. For each $\hat{X}_l \in S_j^+(t_k)$, the time interval that robot j occupies the region $\hat{X}_l$ is given by $\Gamma(\boldsymbol{p}_j^+([t_k, t_j^{fl+}]), \hat{X}_l)$. Supposing that $\Gamma(\boldsymbol{p}_j^+([t_k, t_j^{fl+}]), \hat{X}_l) = \cup_{l=1}^{\hat{m}}[\hat{a}_l, \hat{b}_l)$, where $\hat{m}$

is the number of disjoint intervals in $\Gamma(\boldsymbol{p}_j^+([t_k, t_j^{fl+}]), \hat{X}_l)$. Denote by $\mathcal{T}_j^+(\hat{X}_l)$ the time interval that robot j reserves the area $Res_j(\hat{X}_l)$ according to the updated position trajectory $\boldsymbol{p}_j^+([t_k, \infty))$. Similarly to (33), it can be overapproximated by $\mathcal{T}_j^+(\hat{X}_l) := \cup_{l=1}^{\hat{m}}[\hat{a}_l, \hat{b}_l + T_i^{\text{br}}]$.

Then, the trajectory planning problem (TPP) can be formulated as follows:

$$\text{find} \quad \boldsymbol{\xi}_i([t_k, \infty)) \tag{35a}$$

subject to

$$(7) \text{ and } (8) \tag{35b}$$

$$\boldsymbol{\xi}_i([0, t_k] \cup [t_k, \infty)) \models \varphi_i \tag{35c}$$

$$\psi_i(\text{proj}_l(\boldsymbol{\xi}_i([t_k, \infty)))) \subset \mathbb{F} \tag{35d}$$

$$\psi_i(\text{proj}_l(\xi_i(t))) \cap Res_j(\hat{X}_l) = \emptyset, \ t \in \mathcal{T}_j^+(\hat{X}_l)$$

$$\forall j \in \mathcal{Y}_i(t_k), \forall \hat{X}_l \in S_j^+(t_k). \tag{35e}$$

Constraints (35d) and (35e) guarantee, respectively, that there will be no robot-obstacle collisions and interrobot collisions for robot i.

When relatively simple specifications, e.g., reach-avoid type of tasks, are considered, various existing optimization toolboxes, e.g., IPOPT [33], ICLOCS2 [34], and algorithms, e.g., the configuration space-time search [35], the Hamilton–Jacobian reachability-based trajectory planning [36], RRT^X [37], and the fast robot motion planner [38] can be utilized to solve (35). However, if the specifications are complex LTL formulas, the constraint (35c) is not easy to be verified online using the methods mentioned previously (the PBA $\mathcal{P}_i$ can be used to verify (35c), however, it can not deal with the spatial-temporal collision avoidance constraint (35e) at the same time). Recently, an online RRT-based algorithm is proposed in [39] to generate local paths that guarantee the satisfaction of the global specification. Motivated by this article, an online motion replanning structure is proposed in this article, which contains a local and a global trajectory generation algorithms.

Before proceeding, the following notations are required. Denote by

$$NI_i(t_k) := \{Res_j(\hat{X}_l), \mathcal{T}_j^+(\hat{X}_l)$$

$$\forall \hat{X}_l \in S_j^+(t_k), \forall j \in \mathcal{Y}_i(t_k)\} \tag{36}$$

the set with respect to robot i, which contains all local trajectory information of higher priority neighbors at time t_k. Given a state $s_i \in \hat{\mathbb{X}}_i$, define the function $B_i : \hat{\mathbb{X}}_i \to 2^{S_i}$ as a map from a state $s_i \in \hat{\mathbb{X}}_i$ to a subset of valid Büchi states which correspond to s_i (i.e., $B_i(s_i)) \subseteq \beta_{\mathcal{P}_i}(s_i)$, where $\beta_{\mathcal{P}_i}(s_i)$ is defined in (5)). Function B_i is used to capture the fact that given a partial trajectory, not all Büchi states in $\beta_{\mathcal{P}_i}$ are valid. At the task activation time 0, one has $B_i(\xi_i(0)) = \beta_{\mathcal{P}_i}(\xi_i(0))$. During the online implementation, $B_i(\xi_i(t_k)), t_k > 0$ is updated by

$$B_i(\xi_i(t_k)) = \beta_{\mathcal{P}_i}(\xi_i(t_k))$$

$$\cap \{B_i(\xi_i(t_{k-1})) \cup Post''(B_i(\xi_i(t_{k-1})))\} \tag{37}$$

Algorithm 3: *localTrajectoryGeneration.*

Input: $\xi_i(t_k), B_i(\xi_i(t_k)), \text{Post}(B_i(\xi_i(t_k))), \mathcal{P}_i, V_{\mathcal{P}_i}, \mathbb{O},$ and $NI_i(t_k).$

Return: A local transition system $\mathcal{T}_{c,i}^L$ and a leaf node $\xi_i^f.$

1: Initialize $\mathcal{T}_{c,i}^L = (S_i^L, S_{i,0}^L, AP_{\varphi_i}, \rightarrow_{c,i}^L, L_{c,i})$ and
   $\xi_i^f = \emptyset$, where $S_i^L = S_{i,0}^L = \xi_i(t_k)$ and $\rightarrow_{c,i}^L = \emptyset,$
   $St''(\xi_i(t_k)) = 0.$

2: **for** $k = 1, \ldots, N_i^{\max},$

3:   $\xi_s \leftarrow generateSample(SA_i(t_k)),$

4:   $\xi_n \leftarrow nearest(S_i^L, \xi_s),$

5:   Solve the optimization program $\mathcal{P}(\xi_n, \xi_s, \tau_s)$, which
     returns $(\xi_r, u_i^*),$

6:   $B_i(\xi_r) \leftarrow \beta_{\mathcal{P}_i}(\xi_r) \cap \{B_i(\xi_n) \cup Post''(B_i(\xi_n))\},$

7:   **if** $B_i(\xi_r) \neq \emptyset \wedge V_{\mathcal{T}_{c,i}}(\xi_r, B_i(\xi_r)) < \infty,$

8:     $\mathbb{O}_i(t_k) \leftarrow updateObstacle(\mathbb{O}, NI_i(t_k), [\text{St}(\xi_n)\tau_s,$
       $(\text{St}(\xi_n) + 1)\tau_s]),$

9:     **if** $\text{dist}(\text{proj}_l([\xi_n, \xi_r]), \mathbb{O}_i(t_k)) \geq D_i^{\text{br}},$

10:      $S_i^L \leftarrow S_i^L \cup \{\xi_r\}; \rightarrow_{c,i}^L = \rightarrow_{c,i}^L \cup \{\xi_n \xrightarrow{u_i^*} \xi_r\},$

11:      $St''(\xi_s) \leftarrow St''(\xi_n) + 1,$

12:    **end if**

13: **end if**

14: **if** $\text{proj}_l(\xi_r) \notin \mathcal{B}(\text{proj}_l(\xi_i(t_k)), R),$

15:    $k = N_i^{\max} + 1,$

16:    $\xi_i^f \leftarrow \xi_r,$

17: **end if**

18: **end for**

where

$$\text{Post}(B_i(\xi_i(t_{k-1}))) = \cup_{s_i \in B_i(\xi_i(t_{k-1}))} \text{Post}(s_i). \quad (38)$$

Firstly, the local trajectory generation process is outlined in Algorithm 3. At each time instant t_k, Algorithm 3 takes the state $\xi_i(t_k)$ of robot i, $B_i(\xi_i(t_k))$, $\text{Post}(B_i(\xi_i(t_k)))$, the offline computed PBA $\mathcal{P}_i$, potential function $V_{\mathcal{P}_i}$, the set of static obstacles $\mathbb{O}$, and the local trajectory information of higher priority neighbors, i.e., $NI_i(t_k)$ as input. The output is a local CTS $\mathcal{T}_{c,i}^L := (S_i^L, S_{i,0}^L, AP_{\varphi_i}, \rightarrow_{c,i}^L, L_{c,i})$ that is constructed incrementally and a leaf node $\xi_i^f.$

In line 5, the optimization program $\mathcal{P}(\xi_n, \xi_s, \tau_s)$ is given by

$$\min_{u_i \in \mathbb{U}_i} \quad \|\xi_r - \xi_s\| \quad (39a)$$

subject to

$$\xi_i(0) = \xi_n \quad (39b)$$

$$\xi_n + \int_0^t F_i(\xi_i(s), u_i)ds \in \mathbb{X}_i, \forall t \in [0, \tau_s] \quad (39c)$$

$$\xi_r = \xi_n + \int_0^{\tau_s} F_i(\xi_i(s), u_i)ds \quad (39d)$$

and the optimal solution is $u_i^*.$

The root state of $\mathcal{T}_{c,i}^L$ is robot i's state $\xi_i(t_k)$. The function $\text{st} : S_i^L \rightarrow \mathbb{N}$ maps a state $x \in S_i^L$ to the number of time steps needed to reach the root state $\xi_i(t_k)$. Initially, $\mathcal{T}_{c,i}^L$ contains one state $\xi_i(t_k)$ and 0 transitions, i.e., $st''(\xi_i(t_k)) = 0$, and

Algorithm 4: *updateObstacle.*

Input: $\mathbb{O}, NI_i(t_k)$ and a time interval $[t_1, t_2].$

Return: $\mathbb{O}_i(t_k).$

1: $\mathbb{O}_i(t_k) \leftarrow \mathbb{O},$

2: **for** $j \in \mathcal{Y}_i(t_k)$, **do**

3:   **for** $\hat{X}_l \in S_j^+(t_k)$, **do**

4:     **if** $\mathcal{T}_j^+(\hat{X}_l) \cap [t_k + t_1, t_k + t_2] \neq \emptyset$, **then**

5:       $\mathbb{O}_i(t_k) \leftarrow \mathbb{O}_i(t_k) \cup (Res_j(\hat{X}_l) \cap SA_i(t_k)),$

6:     **end if**

7:   **end for**

8: **end for**

the leaf node $\xi_i^f = \emptyset$ (line 1). In each iteration (lines 2–18), a new state ξ_s is generated randomly from the set $SA_i(t_k)$ using the *generateSample* procedure (line 3), where $SA_i(t_k)$ is the sampling area around robot i, given by

$$SA_i(t_k) := \{(p, \zeta) \in \mathbb{X}_i : p \in \mathcal{B}(\text{proj}_l(\xi_i(t_k)), R + \eta)\}$$

where $\eta > 0$ is an offline chosen constant, which guarantees that there exists $s \in SA_i(t_k)$ such that $\text{proj}_l(s) \notin \mathcal{B}(\text{proj}_l(\xi_i(t_k)), R)$. This condition is essential for checking the terminal condition (line 15). The *nearest* function (line 4) is a standard RRT primitive [29], which returns the nearest state in S_i^L to the new sample ξ_s. Then, one further finds, within the set of states that are reachable from ξ_n at time τ_s, the closest one to the new sample ξ_s, i.e., ξ_r, and the corresponding input u_i^* (line 5). Here, τ_s is the sampling interval (the same one used for constructing the transition system $\mathcal{T}_i$ in Section IV-B3. Once ξ_r is obtained, we further compute the subset of valid Büchi states, which correspond to ξ_r, i.e., $B_i(\xi_r)$, according to (37) (line 6). After that, if both conditions $B_i(\xi_r) \neq \emptyset$ and $V_{\mathcal{T}_{c,i}}(\xi_r, B_i(\xi_r)) < \infty$ are satisfied (which guarantees that there exists a path, starting from ξ_r, that reaches a self-reachable accepting state of $\mathcal{P}_i$, recall Remark 1), obstacles that appear during the time interval $[\text{st}(\xi_n)\tau_s, (\text{st}(\xi_n) + 1)\tau_s]$ are added into the workspace using the function *updateObstacles* (see Algorithm 4). Finally, the state ξ_r is added into S_i^L and the transition relation $\xi_n \xrightarrow{u_i^*} \xi_r$ is added into $\rightarrow_{c,i}^L$ if the distance between the line segment $\text{proj}_l([\xi_n, \xi_r])$ and the obstacles is no less than the braking distance of robot i (lines 9 and 10), and then the time step needed for ξ_r to reach the root state $\xi_i(t_k)$ is recorded (line 11). The algorithm is terminated when the local sampling tree reaches the outside of the sensing area of robot i, and the leaf node ξ_i^f is then given by the corresponding state ξ_r (line 14–17).

Remark 6: The complexity of one iteration of Algorithm 3 is the same as for the standard RRT. The functions *generateSample* and *nearest* are standard RRT primitives (one can refer to [29] for more details). The complexity of the *updateObstacle* process (see Algorithm 4) at time t_k is $\mathcal{O}(1)$ since $|\mathcal{Y}_i(t_k)| \leq N - 1$ and $|S_j^+(t_k)| \leq |\hat{Q}(\mathcal{B}(p_j(t_k), R))|, \forall j$, where $|\hat{Q}(\mathcal{B}(p_j(t_k), R))|$ represents the number of regions contained in the sensing area $\mathcal{B}(p_j(t_k), R)$. The computation of $\text{dist}(\text{proj}_l([\xi_n, \xi_r]), \mathbb{O}_i(t_k))$ can be formulated as a convex

---

Algorithm 5: *globalTrajectoryGenration.*

Input: $P_i(\xi_i^f)$ and $\mathcal{P}_i$.
Return: a path $\boldsymbol{\rho}_i$.
1: $p_i^* = \min_{p_i \in P_i(\xi_i^f)}\{V_{\mathcal{P}_i}(p_i)\}$,
2: **if** $p_i^* \in F_{p,i}^*$ **then**
3:    $\boldsymbol{p}_i \leftarrow DijksCycle(\mathcal{P}_i, p_i^*)$,
4: **else**
5:    $\boldsymbol{p}_i \leftarrow DijksTargets(\mathcal{P}_i, p_i^*, F_{p,i}^*)$,
6: **end if**
7: $\boldsymbol{\rho}_i = \mathtt{pj}_{\hat{\mathbb{X}}_i}(\boldsymbol{p}_i)$,

---

optimization problem and solved in $\mathcal{O}(1)$ since there is a limited number of obstacles in $SA_i(t_k)$ and each obstacle is of the form of a convex region. Moreover, the calculations of $B_i(\xi_r)$ and $V_{\mathcal{T}_{c,i}}(\xi_r, B_i(\xi_r))$ are of the complexity of $\mathcal{O}(1)$ since $\mathcal{P}_i, V_{\mathcal{P}_i}$ are computed offline.

After the local CTS $\mathcal{T}_{c,i}^L$ is obtained, we further need to find a path, starting from the leaf node ξ_i^f, that reaches one of the maximal self-reachable accepting states $F_{p,i}^*$ of $\mathcal{P}_i$. Define

$$P_i(\xi_i^f) := \cup_{s_i \in B_i(\mathcal{T}_{c,i}^L)}(\xi_i^f, s_i)$$

as the set of states in the PBA $\mathcal{P}_i$ that correspond to ξ_i^f. Then, the global trajectory generation process is outlined in Algorithm 5. Algorithm 5 takes the set $P_i(\xi_i^f)$ and the PBA $\mathcal{P}_i$ as input. It first finds the state p_i^* in $P_i(\xi_i^f)$ that has the minimum potential (line 1). Then, if $p_i^* \in F_{p,i}^*$, the function $DijksCycle(\mathcal{P}_i, source)$ (defined in [20]) is used to compute a shortest cycle from the *source* state back to itself (line 3); otherwise, the function $DijksTargets(\mathcal{P}_i, source, targets)$ (defined in [20]) is used to compute a shortest path in $\mathcal{P}_i$ from "*source*" state to one of the state belonging to the set "*targets*" (line 5). The required path is then the projection of $\boldsymbol{p}_i$ on the state space of $\mathcal{T}_{c,i}$ (line 7).

It is possible that after the maximum number of iterations (i.e., $N_i^{\max}$), there exists no local path that reaches the outside of the sensing area of robot i. In this case, the TPP (35) is considered infeasible, robot i switches to *Emerg* mode and the braking controller (12) is applied. When robot i is in *Emerg* mode, it will continue monitoring the environment (by updating $\mathcal{T}_{c,i}^L$). Once a feasible local path is found, it will switch back to *Free* mode. The whole motion coordination process is summarized in Algorithm 6.

The following result shows hat safety is guaranteed under all circumstances.

Theorem 1 (: Safety) If the sensing radius of the robots satisfies $R > 2\max_{i \in \mathcal{V}}\{D_i^{\mathrm{br}} + \Delta \max_{p_i \in \mathbb{W}}\{\|\dot{p}_i\|\}\}$, where Δ is the conflict detection interval, then the resulting real-time plan of Algorithm 6 guarantees that there will be no obstacle-robot and interrobot collisions for robot i.

Proof: If the TPP (35) is feasible (robot i will be in *Free* mode after the trajectory planning process), the local trajectory generation algorithm (see Algorithm 3) and the global trajectory generation algorithm (see Algorithm 5) guarantees that the constraints (35d) and (35e) of (35) are satisfied. That is to say, there will be no obstacle-robot and interrobot collisions

---

Algorithm 6: *motionCoordination.*

Input: (offline) $M_i(\hat{X}_l), \forall \hat{X}_l \in \Xi, \mathcal{T}_{c,i}, \mathsf{B}_i, \mathcal{P}_i$, and $V_{\mathcal{P}_i}$.
Return: Real-time plan $\boldsymbol{\xi}_i^+([t_k, \infty)), t_k \geq 0$.
1: Initialize: $\boldsymbol{\xi}_i^-([0, \infty)) \leftarrow \boldsymbol{\xi}_i^0$, $B_i(\xi_i(0)) \leftarrow \beta_{\mathcal{P}_i}(\xi_i(0))$,
   $\mathtt{Post}(B_i(\xi_i(0))) \leftarrow \cup_{s_i \in B_i(\xi_i(0))}\mathtt{Post}(s_i)$, and Robot
   i is in **Free** mode.
2: **While** $t_k > 0$ and φ_i is not completed **do**,
3:    Compute $B_i(\xi_i(t_k))$ and $\mathtt{Post}(B_i(\xi_i(t_k)))$ according
     to (37) and (38),
4:    $\tilde{\mathcal{N}}_i(t_k) \leftarrow conflictDetection()$,
5:    **if** $\tilde{\mathcal{N}}_i(t_k) \neq \emptyset$ **then**
6:      Robot i switches to **Busy** mode,
7:      $\mathcal{Y}_i(t) \leftarrow planningOrderAssignment()$,
8:      $(\mathcal{T}_{c,i}^L, \xi_i^f) \leftarrow localTrajectoryGeneration()$,
9:      **if** $\xi_i^f \neq \emptyset$, **then**
10:        $\boldsymbol{\rho}_i \leftarrow globalTrajectoryGeneration()$,
11:        $\boldsymbol{\xi}_i^+([t_k, \infty)) \leftarrow$
        $DijksTargets(\mathcal{T}_{c,i}^L, \xi_i(t_k), \xi_i^f) \uplus \boldsymbol{\rho}_i$,
12:        Robot i switches to **Free** mode,
13:      **else**
14:        Robot i switches to **Emerg** mode,
15:        $\boldsymbol{u}_i \leftarrow \boldsymbol{u}_i^{\mathrm{br}}(\xi_i(t_k))$,
16:        $\boldsymbol{\xi}_i^+([t_k, \infty)) \leftarrow \boldsymbol{\xi}_i(\xi_i(t_k), \boldsymbol{u}_i, [0, \infty))$,
17:      **end if**
18:    **else**
19:      $\boldsymbol{\xi}_i^+([t_k, \infty)) \leftarrow \boldsymbol{\xi}_i^-([t_k, \infty))$,
20:    **end if**
21: **end while**

---

for robot i. If the TPP (35) is infeasible (robot i will switch to *Emerg* mode), robot i would apply the braking controller $\boldsymbol{u}_i^{\mathrm{br}}$ until it stops. Since $R > 2\max_{i \in \mathcal{V}}\{D_i^{\mathrm{br}} + \Delta \max_{p_i \in \mathbb{W}}\{\|\dot{p}_i\|\}\}$, it guarantees that conflict between any pair of robots (i, j) will be detected at the time that both robot i and j are outside of the braking area of the other. This means that there will be no interrobot collision during the emergency stop process. On the other hand, when constructing $\mathcal{T}_i$, one has $\psi_i(\mathtt{proj}_l(p_i)) \subseteq \mathbb{F}, \forall p_i \in \hat{\mathbb{X}}_i$, i.e., $\mathtt{dist}(p_i, \mathbb{O}) \geq D_i^{\mathrm{br}}, \forall p_i \in \hat{\mathbb{X}}_i$. Moreover, in the replanning process, it also requires that the distance between $\mathtt{proj}_l([\xi_n, \xi_r])$ and static obstacles $\mathbb{O}$ is no less than D_i^{br} (line 10, Algorithm 3). Therefore, there will be no obstacle-robot collision during the emergency stop process. Thus, no collision will occur for the whole process. $\square$

Theorem 2 (: Correctness) Let $\boldsymbol{\xi}_i^{\mathrm{rt}}([0, \infty))$ be the real-time moving trajectory of robot i. Then, $\boldsymbol{\xi}_i^{\mathrm{rt}}([0, \infty)) \models \varphi_i$ if robot i never enters the *Emerg* mode or stays in *Emerg* mode for a finite time.

Proof: If robot i never enters the *Emerg* mode, then one has from Algorithm 6 that robot i never detects a conflict or the local trajectory generation algorithm (see Algorithm 3) is always feasible for robot i. Then, one can conclude that $\boldsymbol{\xi}_i^{\mathrm{rt}}([0, \infty)) \models \varphi_i$. Otherwise, since robot i stays in *Emerg* mode for a finite time, it means that robot i can always switch back to *Free* mode, which implies the existence of a satisfying trajectory.

In the light of Algorithm 6, the real-time moving trajectory $\boldsymbol{\xi}_i^{\text{rt}}([0,\infty)) \models \varphi_i$. $\hfill\square$

Remark 7: Due to the distributed fashion of the solution and the locally available information, the proposed motion coordination strategy is totally scalable in the sense that the computational complexity of the solution is not increasing with the number of robots. In addition, it is straightforward to extend the work to MRS scenarios where unknown static obstacles and moving obstacles are presented.

Remark 8: The workspace discretization, the planning order assignment, and the overapproximation of the braking area and time influence the completeness (the ability to find a solution when one exists) of the proposed solution. To improve completeness, online refinement of the discretization of the workspace, searching the space of the prioritization scheme (e.g., the randomized search approach with hill climbing [40]), and less-conservative approximation of the braking area and time (using the real-time nonposition information) can be utilized. The disadvantage is that the resulting approach will be computationally more complex.

F. Discussion of Livelocks and Deadlocks

In this section, the possibilities for livelocks and deadlocks are discussed. First, the definitions of deadlock and livelock are given.

Definition 11: We say that a robot i is in a *deadlock* if it remains in *Emerg* mode for an indefinite period (stop forever). We say that a robot i is in a *livelock* if it is in *Free* mode (thus, can move in the workspace) but unable to satisfy its specification.

Livelocks never occur. The reason is that according to our motion coordination algorithm (see Algorithm 6), if a robot i finds during execution that its specification is not satisfiable, then it will switch to *Emerg* mode and thus comes to a stop. In addition, it switches back to *Free* mode only when the task becomes satisfiable again. Therefore, there will be no livelocks.

Before discussing deadlocks, the definition of mutual satisfiability is provided. Denote by $\mathcal{P}_G := \mathcal{P}_1 \otimes \ldots \otimes \mathcal{P}_{|\mathcal{V}|}$ the global PBA, which is the product of all local PBAs $\mathcal{P}_i, i \in \mathcal{V}$.

Definition 12: We say that the set of LTL specifications $\{\varphi_1, \ldots, \varphi_{|\mathcal{V}|}\}$ is *mutually satisfiable* if there exists an accepting run $\boldsymbol{p}_G$ of $\mathcal{P}_G$ such that

$$\phi_i(\texttt{proj}_l(\xi_i(t))) \cap \phi_j(\texttt{proj}_l(\xi_j(t))) = \emptyset$$

$$\forall i,j \in \mathcal{V}, i \neq j, \forall t \geq 0$$

where $\boldsymbol{\xi}_i$ is the projection of $\boldsymbol{p}_G$ onto the local CTS $\mathcal{T}_i$.

When the set of LTL specifications $\{\varphi_1, \ldots, \varphi_{|\mathcal{V}|}\}$ is not mutually satisfiable, deadlocks can happen because the LTL specifications of two or multiple robots cannot be satisfied simultaneously in a collision-free manner. This type of deadlocks is unresolvable. When the set of LTL specifications $\{\varphi_1, \ldots, \varphi_{|\mathcal{V}|}\}$ is mutually satisfiable, deadlocks can also happen due to the incompleteness of the motion coordination algorithm (see Remark 8) or the locally available information of the neighboring robots. Existing methods such as path refinement [41] and simultaneous trajectory planning [42] (instead of the sequential trajectory planning used in this article), can be invoked to resolve this type
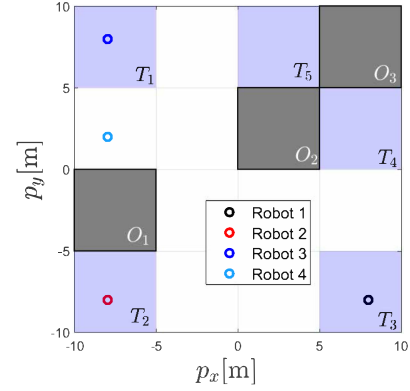

Fig. 4. Workspace for the group of robots in Example 1.

of deadlocks. To detect a possible deadlock, one can *a priori* specify a maximum time $t_{\max}$ allowed for a robot to stay in *Emerg* mode.

V. Case Studies

In this section, two case studies are provided to illustrate the effectiveness and computational tractability of the proposed framework. All simulations are carried out in MATLAB 2018b on a Dell laptop with Windows 10, Intel i7-6600 U CPU2.80 GHz and 16.0 GB RAM.

A. Example 1

Consider a MRS consisting of $N = 4$ robots, the dynamics of robot i is given by (17), and the velocity and input constraints are given by (18). For robots $i \in \{1, 2\}$, one has that $v_{i,\max} = 1\text{m/s}$, $\omega_{i,\max} = 0.5\text{rad/s}$, and $a_{i,\max} = 2\text{m/s}^2$. For robots $i \in \{3, 4\}$, one has that $v_{i,\max} = 1\text{m/s}$, $\omega_{i,\max} = 0.5\text{rad/s}$, and $a_{i,\max} = 1.5\text{m/s}^2$. Then, using the braking controller (21), one can get that $T_i^{\text{br}} = 0.5\text{s}, D_i^{\text{br}} = 0.25\text{m}, i = 1, 2$ and $T_i^{\text{br}} = 0.5\text{s}, D_i^{\text{br}} = 0.33\text{m}, i = 3, 4$. The sensing radius of each robot is $R = 3.5\text{m}$ and the conflict detection period is $\Delta = 0.1\text{s}$. Then, one has that $R > 2\max_{i \in \mathcal{V}}\{D_i^{\text{br}} + \Delta v_{i,\max}\}$.

As shown in Fig. 4, the workspace $\mathbb{W}$ we consider is a $20 \times 20\text{m}^2$ square, where the gray areas, marked as O_1, O_2, O_3, represent three static obstacles, the colored small circles represent the initial position of the robots, and the light blue areas, marked as $T_1, T_2, \ldots, T_5$, represent a set of target regions in the workspace. Initially, $\theta_i = 0, v_i(0) = 0, \forall i$.

Each robot is assigned to persistently survey two of the target regions in the workspace. In LTL formulas, the specification for each robot is given by the following:
1) $\varphi_1 = \square(\mathbb{W} \wedge \neg \mathbb{O}) \wedge \square\diamond T_1 \wedge \square\diamond T_2$;
2) $\varphi_2 = \square(\mathbb{W} \wedge \neg \mathbb{O}) \wedge \square\diamond T_1 \wedge \square\diamond T_5$;
3) $\varphi_3 = \square(\mathbb{W} \wedge \neg \mathbb{O}) \wedge \square\diamond T_2 \wedge \square\diamond T_4$;
4) $\varphi_4 = \square(\mathbb{W} \wedge \neg \mathbb{O}) \wedge \square\diamond T_3 \wedge \square\diamond T_5$;

Where $\mathbb{O} = O_1 \cup O_2 \cup O_3$. First, a rough cell decomposition is given over the workspace $\mathbb{W}$ such that Assumption 1 is satisfied. Then, for each cell $X_l \subset \mathbb{W} \setminus \mathbb{O}$, a grid representation with grid size 0.5m is implemented as a finer workspace discretization. The NBA $\mathbf{B}_i$ associated with $\varphi_i, \forall i$ has 3 states and 8 edges by [24]. The CTS $\mathcal{T}_i$ and the PBA $\mathcal{P}_i$ for each robot
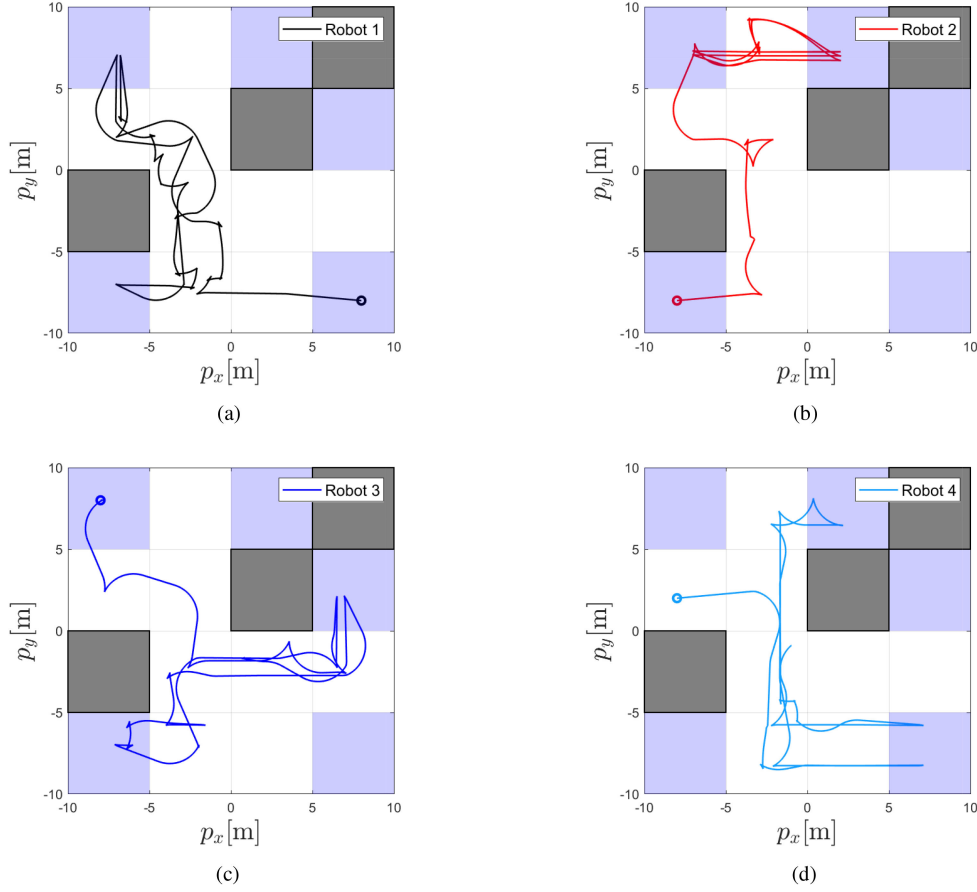
(a)

(b)

(c)

(d)

Fig. 5. Real-time position trajectory of each robot, where the small circle represents the initial position of each robot.

are constructed using LTLCon toolbox [43]. Finally, the map M_i [defined in (27)] and potential function for each $\mathcal{P}_i$ (see Definition 6) are computed.

Initially, each robot i finds a safely satisfying trajectory $\boldsymbol{\xi}_i^0$ (in the form of prefix-suffix). Then, the motion coordination algorithm (see Algorithm 6) is implemented for each robot during online execution. The real-time position trajectory of each robot is depicted in Fig. 5(a)–(d), respectively. It can be seen that during the simulation time $[0\,120]$s, the surveillance task of each robot is satisfied at least once. Fig. 6 shows the evolution of the position trajectories of all robots with respect to time. One can see that the real-time position trajectories are collision free. During the simulation time, conflicts are detected in total 11 times. Whenever conflicts are detected, the planning order assignment module and the trajectory planning module are activated to resolve the conflicts. In Fig. 7(a)–(d), four cases of local motion replanning are depicted, respectively. In each of the subfigures, the circle, marked as dotted line, represents the sensing area of the robot that replans, the dashed lines represent the planned local position trajectories of the robots, the colored small squares represent its (overapproximated) braking areas, the yellow squares represent the conflict region, and the solid line represents the replanned local position trajectory. A video recording of the real-time implementation can be found:[2]
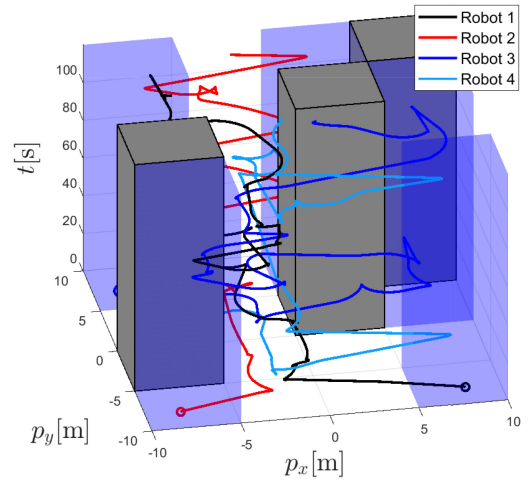
Fig. 6. Evolution of the position trajectories with respect to time.

The real-time evolution of velocity v_i and inputs (a_i, ω_i) of each robot i are plotted in Fig. 8 and Fig. 9, respectively. One can see that the velocity constraints and the input constraints are satisfied by all robots at any time.

Finally, we report the computation times of this example. For the 11 local replannings, the average computation time is 1.51 s and the maximum computation time is 3.30 s. The computation
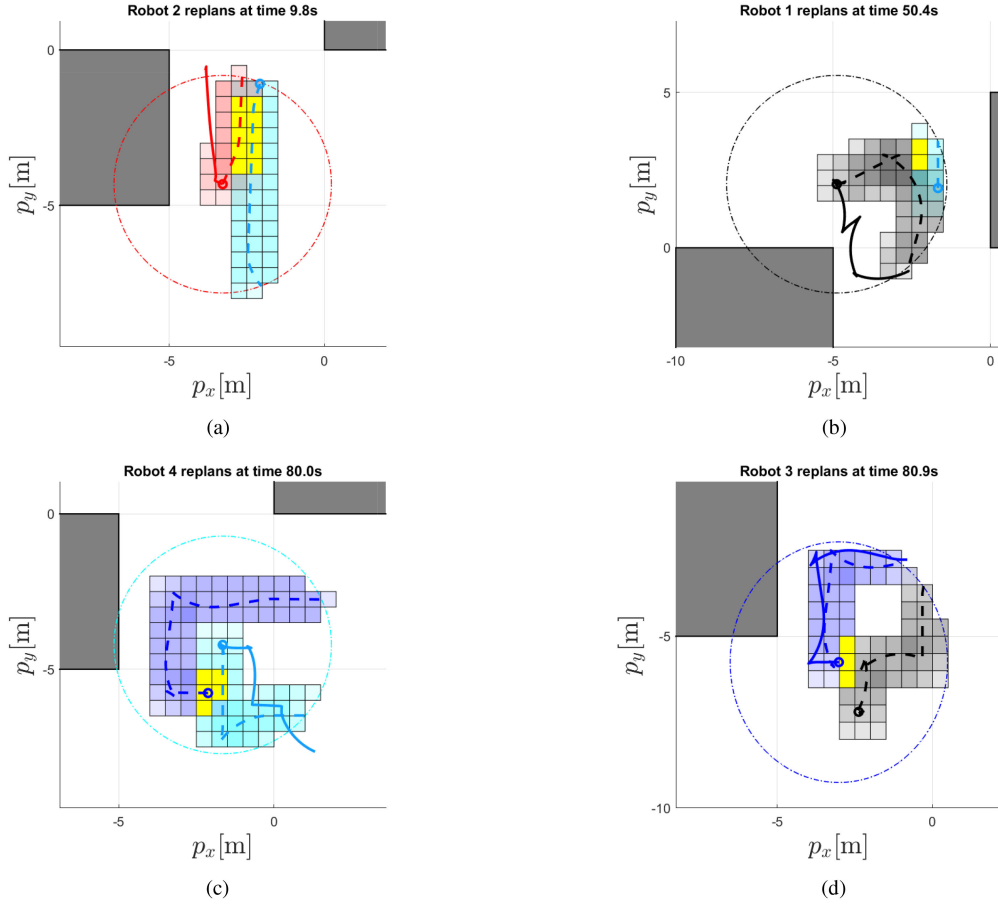
(a)

(b)

(c)

(d)

Fig. 7. Local motion replanning between different conflict robots. The circle, marked with a dotted line, represents the sensing area of the robot that replans, the dashed lines represent the planned local position trajectories of the robots, the colored small squares represent its (overapproximated) braking areas, the yellow squares represent the conflict region, and the solid line represents the replanned local position trajectory.
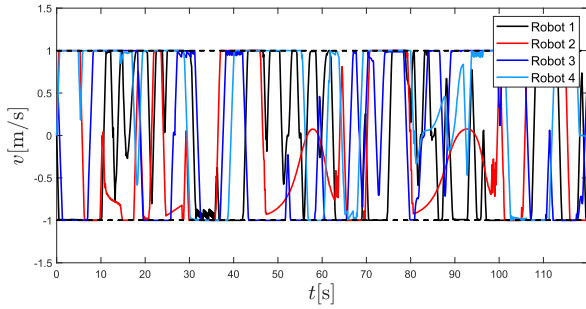
Fig. 8. Real-time evolution of velocity v_i for each robot, where $v_{i,\max} = 1\text{m/s}, i \in \{1, 2, 3, 4\}$.

Fig. 9. Real-time evolution of inputs (a_i, ω_i) for each robot, where $a_{i,\max} = 2\text{m/s}^2, \omega_{i,\max} = 0.5\text{rad/s}, i \in \{1, 2\}$ and $a_{i,\max} = 1.5\text{m/s}^2, \omega_{i,\max} = 0.5\text{rad/s}, i \in \{3, 4\}$.

of the global trajectory (see Algorithm 5) is quite fast (less than 0.1s). Note that after the local and global trajectories are obtained, the toolbox ICLOCS2 [34] is further employed for trajectory smoothness. In addition, it is worth to point out that we have tried 10 different sets of initial states and specifications for the group of robots, and we find that the initial states and the specifications have little effect on the average time and maximum time of local replanning. There is no change of order of magnitude in the computation time for all trials.
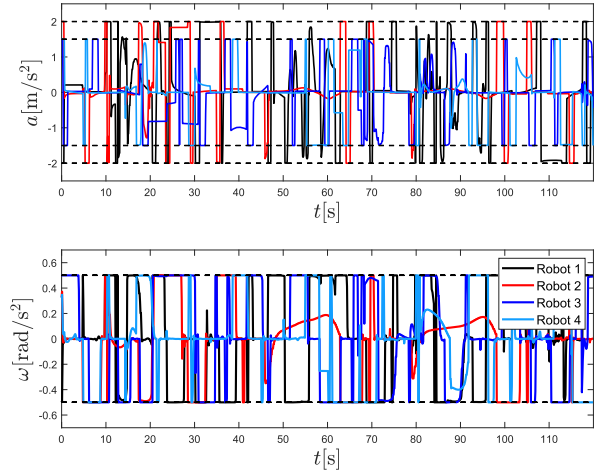
B. Example 2

This example aims at demonstrating the computational tractability of the approach with respect to the number of robots.
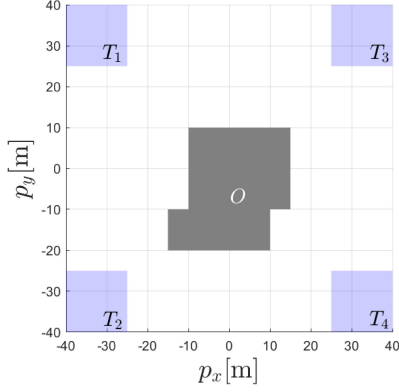
Fig. 10. Workspace for the group of robots in Example 2.

TABLE II
CT, ATLR, AND MTLR WITH RESPECT TO THE NUMBER OF ROBOTS

number of robots	CT	ATLR (s)	MTLR (s)
2	0	-	-
4	6	0.23	2.99
8	19	0.31	3.80
16	72	0.63	4.94

As depicted in Fig. 10, the workspace $\mathbb{W}$ considered now is a $80 \times 80 \text{m}^2$ square, where the gray area, marked as O is the obstacle, and the light blue areas, marked as $T_1, T_2, \ldots, T_4$, represent four target regions in the workspace. We consider, respectively, the cases where 2, 4, 8, and 16 robots are working in the workspace. The dynamics of robot i is given by a double integrator, i.e.,

$$\dot{p}_i = v_i$$
$$\dot{v}_i = u_i, \forall i$$

where $p_i, v_i, u_i \in \mathbb{R}^2$ represent the position, velocity, and input of robot i, respectively. The velocity and input of robot i are subject to the hard constraints $||v_i(t)|| \leq v_{i,\max} = 3\text{m/s}, ||u_i(t)|| \leq u_{i,\max} = 6\text{m/s}^2, \forall i$. The braking controller $\boldsymbol{u}_i^{\text{br}}$ is designed as

$$u_i^{\text{br}}(t) = \begin{cases} -u_{i,\max} \frac{v_i(t)}{||v_i(t)||}, & \text{if } ||v_i(t)|| \neq 0 \\ 0_2, & \text{if } ||v_i(t)|| = 0. \end{cases}$$

Then, one can derive the braking time $T_i^{\text{br}} = ||v_{i,\max}||/||u_{i,\max}|| = 0.5\text{s}$ and the braking distance $D_i^{\text{br}} = ||v_{i,\max}||^2/2||u_{i,\max}|| = 0.75\text{m}, \forall i$. The sensing radius of each robot is $R = 6\text{m}$ and the conflict detection period is $\Delta = 0.1\text{s}$.

For simplicity, we consider that half of the robots are assigned to persistently survey target regions T_1 and T_3, and the other half are assigned to persistently survey target regions T_2 and T_4. In LTL formulas, the specification for each robot is given by the following:

1) $\varphi_i = \Box(\mathbb{W} \wedge \neg O) \wedge \Box\Diamond T_1 \wedge \Box\Diamond T_4, i = 1, 3, \ldots, 15$;
2) $\varphi_i = \Box(\mathbb{W} \wedge \neg O) \wedge \Box\Diamond T_2 \wedge \Box\Diamond T_3, i = 2, 4, \ldots, 16$.

A grid representation with grid size 2m is implemented as the workspace discretization.

For each of the cases (2, 4, 8, or 16 robots), the simulation time is 150s. The simulation results are summarized in Table II,

where the total number of conflict times (CT), the average computation time for local replanning (ATLR), and the maximum computation time for local replanning (MTLR) are reported. It can be seen that as the number of robots grows (exponentially), the CT grows significantly (the reason is that the workspace is the same). The ATLR and MTLR grow, but not significantly. One can see that even when there are 16 robots, the local replanning is still efficient.

VI. CONCLUSION

In this article, the online MRMC problem for a group of mobile robots moving in a shared workspace was considered. Under the assumptions that each robot has only local view and local information, and subject to both state and input constraints, a fully distributed motion coordination strategy was proposed for steering individual robots in a common workspace, where each robot is assigned a LTL specification. It was shown that the proposed strategy can guarantee collision-free motion of each robot. In the future, MRMC under both team and individual tasks will be of interest. In addition, the experimental validation of the proposed strategy by real-world robots will be pursued.

REFERENCES

[1] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *Int. J. Adv. Robot. Syst.*, vol. 10, no. 12, pp. 1–18, 2013.

[2] L. E. Parker, "Path planning and motion coordination in multiple mobile robot teams," in *Encyclopedia of complexity and system science*. New York, NY, USA: Springer, 2009, pp. 5783–5800.

[3] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Auton. Robot Veh.*, Springer, 1986, pp. 396–404.

[4] D. Panagou, "Motion planning and collision avoidance using navigation vector fields," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 2513–2518.

[5] L. Gracia, F. Garelli, and A. Sala, "Reactive sliding-mode algorithm for collision avoidance in robotic systems," *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 6, pp. 2391–2399, Nov. 2013.

[6] H. Farivarnejad, S. Wilson, and S. Berman, "Decentralized sliding mode control for autonomous collective transport by multi-robot systems," in *Proc. IEEE 55th Conf. Decis. Control*, 2016, pp. 1826–1833.

[7] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Trans. Robot.*, vol. 33, no. 3, pp. 661–674, Jun. 2017.

[8] P. Glotfelter, J. Cortés, and M. Egerstedt, "Nonsmooth barrier functions with applications to multi-robot systems," *IEEE control Syst. Lett.*, vol. 1, no. 2, pp. 310–315, Oct. 2017.

[9] L. Pallottino, V. G. Scordio, A. Bicchi, and E. Frazzoli, "Decentralized cooperative policy for conflict resolution in multivehicle systems," *IEEE Trans. Robot.*, vol. 23, no. 6, pp. 1170–1183, Dec. 2007.

[10] G. A. Bekey, *Autonomous Robots: From Biological Inspiration to Implementation and Control*. Cambridge, MA, USA: MIT Press, 2005.

[11] K. Azarm and G. Schmidt, "Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation," in *Proc. Int. Conf. Robot. Autom.*, 1997, vol. 4, pp. 3526–3533.

[12] Y. Guo and L. E. Parker, "A distributed and optimal motion planning approach for multiple mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom. (Cat. No 02CH37292)*, 2002, vol. 3, pp. 2612–2619.

[13] W. Sheng, Q. Yang, J. Tan, and N. Xi, "Distributed multi-robot coordination in area exploration," *Robot. Auton. Syst.*, vol. 54, no. 12, pp. 945–955, 2006.

[14] C. Liu, C.-W. Lin, S. Shiraishi, and M. Tomizuka, "Distributed conflict resolution for connected autonomous vehicles," *IEEE Trans. Intell. Veh.*, vol. 3, no. 1, pp. 18–29, Mar. 2017.

[15] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimality and robustness in multi-robot path planning with temporal logic constraints," *Int. J. Robot. Res.*, vol. 32, no. 8, pp. 889–911, 2013.

[16] Y. Shoukry *et al.*, "Linear temporal logic motion planning for teams of underactuated robots using satisfiability modulo convex programming," in *Proc. IEEE 56th Annu. Conf. Decis. Control*, 2017, pp. 1132–1137.

[17] Y. E. Sahin, P. Nilsson, and N. Ozay, "Multirobot coordination with counting temporal logics," *IEEE Trans. Robot.*, vol. 36, no. 4, pp. 1189–1206, Aug. 2020.

[18] Y. Kantaros and M. M. Zavlanos, "STyLuS*: A. temporal logic optimal control synthesis algorithm for large-scale multi-robot systems," *Int. J. Robot. Res.*, vol. 39, no. 7, pp. 812–836, 2020.

[19] J. Alonso-Mora, J. A. DeCastro, V. Raman, D. Rus, and H. Kress-Gazit, "Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles," *Auton. Robots*, vol. 42, no. 4, pp. 801–824, 2018.

[20] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," *Int. J. Robot. Res.*, vol. 34, no. 2, pp. 218–235, 2015.

[21] P. Yu and D. V. Dimarogonas, "A fully distributed motion coordination strategy for multi-robot systems with local information," in *IEEE Conf. American Control*, pp. 1423–1428, 2020, *arXiv:2004.10437*i) .

[22] C. Baier and J.-P. Katoen, *Principles of Model Checking*. Cambridge, MA, USA: MIT press, 2008.

[23] J. R. Büchi, "On a decision method in restricted second order arithmetic," in *The Collected Works of* J. Richard Büchi. New York, NY, USA: Springer, 1990, pp. 425–435.

[24] P. Gastin and D. Oddoux, "Fast LTL to Büchi automata translation," in Proc. *Int. Conf. Comput. Aided Verification*, Springer, 2001, pp. 53–65.

[25] C. I. Vasile and C. Belta, "Sampling-based temporal logic path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 4817–4822.

[26] E. A. Emerson, "Temporal and Modal Logic," *Formal Models and Semantics*. New York, NY, USA: Elsevier, 1990, pp. 995–1072.

[27] J.-C. Latombe, *Robot Motion Planning*, vol. 124, Berlin, Germany: Springer Science & Business Media, 2012.

[28] S. Bhattacharya, V. Kumar, and M. Likhachev, "Search-based path planning with homotopy class constraints." in *Proc. AAAI Conf. Artif. Intell.*, 2010, pp. 1230–1237.

[29] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," *Algorithmic Comput. Robot. New Directions*, no. 5, pp. 293–308, 2001.

[30] M. M. Quottrup, T. Bak, and R. Zamanabadi, "Multi-robot planning: A timed automata approach," in *Proc. IEEE Int. Conf. Robot. Autom.*, *Proc.*, 2004, vol. 5, pp. 4417–4422.

[31] T. M. Howard, C. J. Green, and A. Kelly, "Receding horizon model-predictive control for mobile robot navigation of intricate paths," in *Field and Service Robotics*. New York, NY, USA: Springer, 2010, pp. 69–78.

[32] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization." in *Robotics: Science and Systems*, vol. 9. New York, NY, USA: Citeseer, 2013, pp. 1–10.

[33] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization," *Comput. Chem. Eng.*, vol. 33, no. 3, pp. 575–582, 2009.

[34] Y. Nie, O. Faqir, and E. C. Kerrigan, "ICLOCS2: Solve your optimal control problems with less pain," *IFAC Secretariat*, 2018.

[35] D. Parsons and J. Canny, "A motion planner for multiple mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1990, pp. 8–13.

[36] M. Chen, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Robust sequential trajectory planning under disturbances and adversarial intruder," *IEEE Trans. Control Syst. Technol.*, vol. 27, no. 4, pp. 1566–1582, Jul. 2019.

[37] M. Otte and E. Frazzoli, "RRT^X: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *Int. J. Robot. Res.*, vol. 35, no. 7, pp. 797–822, 2016.

[38] H.-C. Lin, C. Liu, and M. Tomizuka, "Fast robot motion planning with collision avoidance and temporal optimization," in *Proc. 15th Int. Conf. Control, Autom., Robot. Vis.*, 2018, pp. 29–35.

[39] C. I. Vasile and C. Belta, "Reactive sampling-based temporal logic path planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 4310–4315.

[40] M. Bennewitz, W. Burgard, and S. Thrun, "Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots," *Robot. Auton. Syst.*, vol. 41, no. 2-3, pp. 89–99, 2002.

[41] M. G. Atia, O. Salah, and H. Ei-Hussieny, "OGPR: An obstacle-guided path refinement approach for mobile robot path planning," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, 2018, pp. 844–849.

[42] K. Molloy, L. Denarie, M. Vaisset, T. Siméon, and J. Cortés, "Simultaneous system design and path planning: A sampling-based algorithm," *Int. J. Robot. Res.*, vol. 38, no. 2/3, pp. 375–387, 2019.

[43] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Trans. Autom. Control*, vol. 53, no. 1, pp. 287–297, 2008.

Pian Yu (Graduate Student Member) received the M.Sc. degree in control science and control engineering from Wuhan University, Wuhan, China, in 2016, and the Ph.D. degree in electrical engineering from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2021.

She is currently a Postdoctoral Researcher with the Division of Decision and Control Systems, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology. Her main research interest includes multiagent systems, motion planning and coordination, event-triggered control, and formal methods.

Dimos V. Dimarogonas (Senior Member, IEEE) received the Diploma degree in electrical and computer engineering and the Ph.D. degree in mechanical engineering from the National Technical University of Athens, Athens, Greece, in 2001 and 2007, respectively.

From May 2007 to February 2009, he was a Postdoctoral Researcher with the Automatic Control Laboratory, School of Electrical Engineering, KTH Royal Institute of Technology, Stockholm, Sweden, and a Postdoctoral Associate with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, USA. He is currently a Professor with Division of Decision and Control Systems, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology. His current research interests include multiagent systems, hybrid systems, robot navigation, networked control, and event-triggered control.

He serves on the Editorial Board of *Automatica* and the IEEE TRANSACTIONS ON CONTROL OF NETWORK SYSTEMS. He received an ERC Starting Grant in 2014, an ERC Consolidator Grant in 2019, and was awarded a Wallenberg Academy Fellow Grant in 2015.