Priority-Based Distributed Coordination for Heterogeneous Multi-Robot Systems With Realistic Assumptions

Michele Cecchi, Matteo Paiano, Anna Mannucci, *Member, IEEE*, Alessandro Palleschi, *Graduate Student Member, IEEE*, Federico Pecora, and Lucia Pallottino, *Senior Member, IEEE*

***Abstract*—A standing challenge in current intralogistics is to reliably, effectively, yet safely coordinate large-scale, heterogeneous multi-robot fleets without posing constraints on the infrastructure or unrealistic assumptions on robots. A centralized approach, proposed by some of the authors in prior work, allows to overcome these limitations with medium-scale fleets (i.e., tens of robots). With the aim of scaling to hundreds of robots, in this article we explore a decentralized variant of the same approach. The proposed framework maintains the key features of the original approach, namely, ensuring safety despite uncertainties on robot motions, and generality with respect to robot platforms, motion planners and controllers. We include considerations on liveness and report solutions to prevent or recover from deadlocks in specific situations. We validate the approach empirically in simulation with large, heterogeneous multi-robot fleets (with up to 100 robots) operating in both benchmark and realistic environments.**

***Index Terms*—Distributed robot systems, multi-robot systems, planning, scheduling and coordination.**

I. Introduction

PIONEERED by Amazon and accelerated by ever-increasing e-commerce demand, Autonomous Mobile Robots (AMRs) have seen increasing uptake over the past years [1] as a cost-effective way to automate material handling [2]. The increased popularity of AMRs poses new challenges in efficiently and safely managing large heterogeneous fleets, in which robots may differ in dimensions, shapes, dynamic constraints, and capabilities [3]. This coordination problem has been tackled by the scientific community with both centralized and distributed approaches [3]. In centralized approaches, a single decision-making entity collects global information on the fleet (e.g., tasks, positions, and paths of all robots) and updates all robot actions accordingly. Global properties, such as safety and liveness guarantees or performance optimization, can be enforced by reasoning about the fleet as a whole but may require exponential computation [4]. Therefore, to scale at large fleets, these methods often pose constraints on the infrastructure, robot kinodynamics, geometries, controllers, or all of the above. A centralized approach proposed by some of the authors in [5] has shown that it is possible to scale to tens of robots without imposing these unrealistic assumptions. However, communication uncertainty and/or real-time constraints may limit the ability of maintaining up-to-date snapshots of the fleet status [5], or of committing global plans [4]. To overcome these limitations, research has focused on improving wireless technologies towards ultra-reliable, low-latency communications [6]; co-optimizing robot motions and communication constraints [5], [7], [8]; developing coordination algorithms with relaxed temporal requirements on communications [5], [9], [10]; decentralizing coordination to let robots autonomously decide their future actions based on local information and communication with nearby robots [11]. Thanks to locality, in fact, decentralized approaches are less prone to communication uncertainty. These methods can be made robust to lack of global information and central unit faults [11], and scale more easily to large fleets. However, they are less able to enforce liveness or optimality. Different decentralized solutions have been proposed in the literature, based on, e.g., resource-sharing protocols and re-planning strategies [12]; autonomous motion planning and conflict resolution based on removal strategies and private zone mechanisms [13]; combination of optimal control with model-based heuristics [14]; online planning order assignment and a multi-step motion planning process [15]. Most of these methods require the robots to travel along predefined paths and/or the installation of additional infrastructure into the working environment (thus limiting flexibility and reconfigurability).

In this paper, we propose a novel distributed coordination algorithm that targets applications where motion planning, coordination and control are "loosely-coupled" [10], that is, the robots

Manuscript received March 1, 2021; accepted June 6, 2021. Date of publication June 21, 2021; date of current version July 5, 2021. This letter was recommended for publication by Associate Editor Q. Chang and Editor J. Yi upon evaluation of the reviewers' comments. This work has received funding from the European Union's Horizon 2020 Research and Innovation Program under agreement no. 732737 (ILIAD), by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Department of Excellence), by Vinnova under project AutoHauler, and by the Semantic Robots KKS research profile. *(Michele Cecchi and Matteo Paiano contributed equally to this work. Corresponding author: Anna Mannucci.)*

Michele Cecchi, Matteo Paiano, Alessandro Palleschi, and Lucia Pallottino are with the Research Center "E. Piaggio," and Dipartimento di Ingegneria dell'Informazione, University of Pisa, Italy (e-mail: m.cecchi13@studenti.unipi.it; paianomatteo@libero.it; alessandro.palleschi@phd.unipi.it; lucia.pallottino@unipi.it).

Anna Mannucci and Federico Pecora are with the Centre for Applied Autonomous Sensor Systems (AASS), Örebro University, Sweden (e-mail: anna.mannucci90@gmail.com; federico.pecora@oru.se).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2021.3091016, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3091016

share a common environment, and may be asynchronously assigned online-posted individual goals. We build our work upon the centralized, priority-based approach to multi-robot coordination proposed in [5] since validated formally and supported by several projects with industrial partners, including Scania, Epiroc, Bosch and Volvo [16]. Here, we exploit local inter-robot communication to let robots 1) autonomously identify and revise over time some of the future conflicting configurations along their paths, and 2) autonomously coordinate their motions via dynamic, heuristic-based precedence constraints. Our approach has several unique properties: 1) robot motions and communications are asynchronous; 2) no particular requirement is imposed on motion planners, e.g., paths can be computed on the fly by each robot, or be extracted from a roadmap known to all; 3) minimal requirements are imposed on robot controllers, which only have to ensure bounded spatial tracking errors and the ability to commit to dynamically feasible set point updates; 4) the amount of information shared among the robots is minimized; 5) robot precedence constraints can be dynamic and user-specified, and are functions of time — hence robots can follow each other along shared regions of the environment; 6) safety is guaranteed, under the assumptions of reliable communication and conservative kinodynamic models; 7) the approach scales to large multi-robot fleets (tested with fleets of up to 100 robots) while outperforming the original centralized method.

II. Problem Formulation

We consider a multi-robot fleet $\mathcal{R} = \{r_i\}_{i=1}^{N}$ of possibly heterogeneous robots[1], each of which is associated with a unique $\mathrm{ID}_i \in \mathbb{N}$ and a (possibly dynamic) priority $\pi_i \in \mathbb{N}$. Each robot navigates in a bounded obstacle-free environment $\mathcal{W}^{\text{free}} \in \mathbb{R}^2$ according to a conservative kinodynamic model $g_i(q_i, \dot{q}_i, \ddot{q}_i, \nu_i) = 0$, $q_i \in \mathcal{Q}_i$, $\nu_i \in [\underline{\nu}_i, \bar{\nu}_i]$, with $\mathcal{Q}_i$ and $[\underline{\nu}_i, \bar{\nu}_i]$ being the robot configuration and control space, respectively. Let $W_i : \mathcal{Q}_i \to \mathbb{R}^2$ be mapping of robot r_i's configurations to Cartesian space. Also, let $R_i(q_i) \subset \mathbb{R}^2$ be r_i's collision space when placed in configuration $q_i \in \mathcal{Q}_i$ and $\mathcal{Q}_i^{\text{free}} = \{q_i \in \mathcal{Q}_i | R_i(q_i) \subseteq \mathcal{W}^{\text{free}}\}$. The symbol $\mathrm{circ}(G)$ indicates the radius of the circle circumscribing the geometry $G \subset \mathbb{R}^2$. Robots are not required to be synchronized on a common Coordinated Universal Time (UTC). Henceforth, the symbols t and t_i refer to the continuous global time and the robot local time, respectively. The robots are equipped with an on-board, omnidirectional, range-limited communication device which can be used for coordination. We define r_i's neighborhood $\mathcal{N}_i(q_i, t)$ as the set of all robots within its communication range $\mathcal{S}_i(q_i)$ at time t, that is, $\mathcal{N}_i(q_i, t) = \{r_j \in \mathcal{R} \setminus \{r_i\} | W_j(q_j(t)) \in \mathcal{S}_i(q_i)\}$[2]. We assume all robots have the same communication radius ρ (see Fig. 2a). Thus, $r_i \in \mathcal{N}_j(t) \iff r_j \in \mathcal{N}_i(t)$ for all t and for all pairs $(r_i, r_j) \in \mathcal{R}^2$, $i \neq j$ (i.e., communication is symmetric).

Each idle robot may be assigned an individual task, possibly asynchronously posted, which involves moving from its current configuration $q_i^{\text{s}} \in \mathcal{Q}_i^{\text{free}}$ to a target configuration $q_i^{\text{g}} \in \mathcal{Q}_i^{\text{free}}$. We assume the set of robot targets $\{q_i^{\text{g}}\}_{i=1}^{N}$ to be *well-posed* at each time, that is, the multi-robot trajectory planning problem admits at least one feasible solution [17]. The objective is to define a coordination algorithm which leverages local inter-robot communication to compute and revise the robot trajectories so that: (O1) collisions between robots never happen (*safety*); (O2) all robots achieve their destination in finite time (*liveness*). The proposed algorithm should be (O3) general to robots and robust to uncertainties in trajectory execution.

[1]We use the notation x_i when the variable x is related to robot r_i, and $\underline{x}$, $\bar{x}$, $\tilde{x}$ to indicate lower/upper bounds and estimated values, respectively.

[2]The dependency of $\mathcal{N}_i(q_i, t)$ on q_i will be omitted in the following.

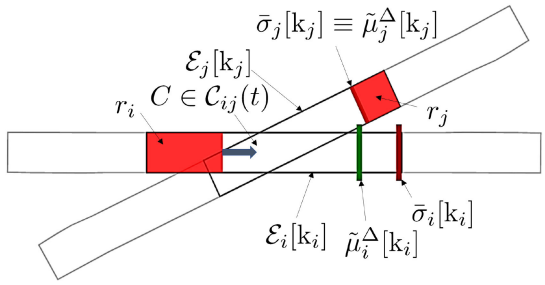


Fig. 1. Preliminary definitions for coordination purposes.

III. Method

The centralized approach in [4], [5] and [10] fulfills O1–O3 by relying on decoupled motion planning and heuristically-revised precedence constraints to regulate access to and progress through so-called *critical sections*, i.e., pairwise contiguous overlapping configurations along the robots' paths. In this paper, we substitute the centralized algorithm with a local coordination algorithm running on each robot.

A. Definition

We will use the symbols T_i^c and $\mathrm{k}_i \in \mathbb{N}$ to indicate each r_i's control period and discrete time. We assume each T_i^c is constant; however, different robots may have different control periods. Henceforth, let $[\mathrm{k}_i]$ indicate the time interval $t \in [\mathrm{k}_i T_i^c, (\mathrm{k}_i + 1)T_i^c]$ and $[\mathrm{k}_i] \cap [\mathrm{k}_j]$ refer to the intersection of time intervals of robots r_i and r_j. All concepts presented in the following paragraph are graphically shown in Fig. 1.

a) Paths: As in [10] and [5], robot paths are computed in a decoupled fashion by motion planners that are private to the robots. We use the symbol $\boldsymbol{p}_i$ to refer to each robot r_i's path; let $\sigma_i \in [0, 1]$ and γ_i be the arc length and the length of the path $\boldsymbol{p}_i$, respectively. Each planning instance succeeds iff $\boldsymbol{p}_i(0) = q_i^{\text{s}}$, $\boldsymbol{p}_i(1) = q_i^{\text{g}}$ and $\boldsymbol{p}_i(\sigma_i) \in \mathcal{Q}_i^{\text{free}}$ for all $\sigma_i \in [0, 1]$. Let $\sigma_i(t)$ indicate the progress of the robot along path $\boldsymbol{p}_i$ at time t.

b) Envelope chunks: To account for spatial uncertainties in localization and control, we define the envelope chunk $\mathcal{E}_i[\mathrm{k}_i]$ as a set of spatial constraints [18] on the future configurations along the path $\boldsymbol{p}_i$ such that

$$\bigcup_{\sigma_i \in [L_i[\mathrm{k}_i], U_i[\mathrm{k}_i]]} R_i(\boldsymbol{p}_i(\sigma_i)) \subseteq \mathcal{E}_i[\mathrm{k}_i] \tag{1}$$

with $L_i[\mathrm{k}_i] = \sigma_i(\mathrm{k}_i T_i^c)$, $U_i[\mathrm{k}_i] = L_i[\mathrm{k}_i] + \delta_i^{\text{h}}$, δ_i^{h} being the chunk horizon. As we will see in Section III-D, the value of δ_i^{h} affects safety and liveness and, thus, should be selected

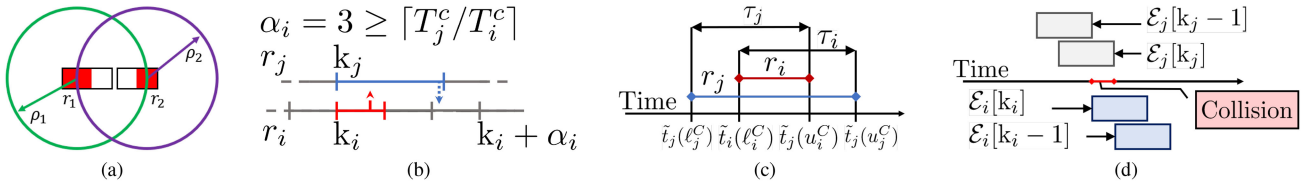


Fig. 2. Concepts of distributed coordination: (a) communication radii; (b) due to asynchronous communication, in the interval $[\mathrm{k}_j]$, robot r_j will ignore r_i's state as communicated in any $t \in [\mathrm{k}_i T_i^c, (\mathrm{k}_i + \alpha_i) T_i^c)$, hence, the information considered by r_j is at most $\alpha_i T_i^c$ seconds old; (c) minimum completion time heuristic; (d) robots may collide if they don't reason about transmission delays.

coupled with the robot's maximum velocity, control period and communication range [9]. We require that $\mathcal{E}_i$ terminates with a contingency maneuver $d_i(q_i, \dot{q}_i, \ddot{q}_i, [\underline{\nu}_i, \bar{\nu}_i], g_i, t)$. Henceforth, for simplicity, we assume that: 1) equality holds in (1) (i.e., the envelope chunk is the sweep of the robot's geometry over some future configurations of its path); 2) the contingency maneuver is a braking maneuver for all robots (which holds, e.g., with car- and unicycle-like vehicles). Approaches based on safety discs [19] may be used to extend the framework to plane-like robots.

c) Critical sections: Let $\mathcal{E}_j[\mathrm{k}_i]$ be the chunk of a neighboring robot r_j received at time k_i. We define a critical section $C \in \mathcal{C}_i[\mathrm{k}_i]$ as the tuple $\langle \ell_i^C, u_i^C, \ell_j^C, u_j^C \rangle$ of continuous intervals of the arc lengths σ_i and σ_j (along $\mathcal{E}_i[\mathrm{k}_i]$ and $\mathcal{E}_j[\mathrm{k}_i]$, respectively) such that for every $\sigma_i \in (\ell_i^C, u_i^C)$, there exists $\sigma_j \in (\ell_j^C, u_j^C)$ such that $R_i(\boldsymbol{p}_i(\sigma_i)) \cap R_j(\boldsymbol{p}_j(\sigma_j)) \neq \emptyset$, and vice versa. Specifically, ℓ_i^C is the highest value of σ_i before robot i enters C, and u_i^C is the lowest value of σ_i after robot i exits C (analogously for j). Critical sections in the set $\mathcal{C}_i[\mathrm{k}_i]$ are ordered with increasing values of ℓ_i^C. Also, we use the symbol $\mathcal{C}_{ij}[\mathrm{k}_i]$ to indicate the (ordered) subset of $\mathcal{C}_i[\mathrm{k}_i]$ involving the pair of robots (r_i, r_j). Remarkably, 1) since communication is *asynchronous*, the pairs $(\mathcal{E}_i[\mathrm{k}_i], \mathcal{E}_j[\mathrm{k}_i])$ and $(\mathcal{E}_i[\mathrm{k}_j], \mathcal{E}_j[\mathrm{k}_j])$ may differ for some $t \in [\mathrm{k}_i] \cap [\mathrm{k}_j]$, that is, $C \in \mathcal{C}_{ij}(t) \not\Rightarrow C \in \mathcal{C}_{ji}(t)$ at all global times t; 2) since communication is *local*, then there may exists a pair $(C, C'), C \in \mathcal{C}_{ij}[\mathrm{k}_i], C' \in \mathcal{C}_{ij}[\mathrm{k}_i + 1]$ such that $C \cap C' \neq \emptyset$, but $C \neq C'$, i.e., entry and exit values ℓ_i^C, u_i^C, ℓ_j^C, u_j^C may change for consecutive k_i.

d) Precedence constraints: A precedence constraint $\langle \ell_i^C, u_j^C \rangle$ is a constraint on the temporal profile $\sigma_i(t)$ such that $\sigma_j(t) < u_j^C \Rightarrow \sigma_i(t) \leq \ell_i^C$, i.e. robot r_i is not allowed to navigate beyond its own arc length ℓ_i^C until robot r_j has reached arc length u_j^C along its path. Since the current chunks and hence sets $\mathcal{C}_i$ are updated over time, robots can follow each other when possible as in [10]. Coordinated decisions on precedence constraints can be used to safely regulate the access and the progress throughout each set $\mathcal{C}_i$. Agreement on precedence orders (i.e., either robot r_i or robot r_j may enter a shared critical section) is sought via *pairwise-shared* heuristic functions h_{ij} (see Section III-D for details). Similarly to our previous work, ordering decisions which may not be communicated on time ([5]) or which may not be physically realizable ([5] and [10]) are recognized by each robot and filtered out (see Alg. 2 for details). This is achieved by forward propagating the robot kinodynamics: we define the *decision point* $\tilde{\mu}_i^{\Delta}[\mathrm{k}_i]$ as the value of σ_i at which the robot may safely stop (if required to) while assuming the robot drives with maximum acceleration for Δ seconds and then applies maximum deceleration till stopping, i.e.,

$$\tilde{\mu}_i^{\Delta}[\mathrm{k}_i] = L_i[\mathrm{k}_i] + \gamma_i^{-1} \left[\int_0^{\Delta} g_i(q_i, \dot{q}_i, \ddot{q}_i, \bar{\nu}_i, t)\, dt + d_i \right]. \quad (2)$$

Specifically, if $\Delta = T_i^c$, then $\tilde{\mu}_i^{\Delta}[\mathrm{k}_i]$ (upper) bounds the last feasible point the robot can be safely required to yield in the current period. Also, as in [5] conservative (upper) bounds of clock de-synchronization (if clocks are synchronized among robots) or network latency (w/o clock synchronization) and sensing-actuation delays may be used to inflate the value Δ of a quantity ζ so as to ensure safety with bounded inter-robot communication delays.

e) Critical points: Let $\mathcal{T}_i[\mathrm{k}_i]$ be r_i's set of precedence constraints $\langle \ell_i^C, u_j^C \rangle$ along the current $\mathcal{E}_i[\mathrm{k}_i]$ which may require robot r_i to yield for a neighboring robot r_j. Let also

$$\alpha_i \triangleq \max_{r_j \in \mathcal{R} \setminus \{r_i\}} \lceil (T_j^c + \zeta) / T_i^c \rceil \quad (3)$$

be the maximum delay between $\mathcal{E}_i[\mathrm{k}_i]$ and the local views $\{\mathcal{E}_i[\mathrm{k}_j]\}_{r_j \in \mathcal{N}_i[\mathrm{k}_i]}$ of neighboring robots caused by asynchronous communication (see Fig. 2b and Section III-D for details). We define the *critical point* $\bar{\sigma}_i[\mathrm{k}_i]$ as

$$\bar{\sigma}_i[\mathrm{k}_i] = \begin{cases} 0 & \mathrm{k}_i \leq \alpha_i, \\ \min\{\underline{\ell}_i^C, U_i[\mathrm{k}_i - \alpha_i]\} & \mathrm{k}_i > \alpha_i, \end{cases}$$

$$\underline{\ell}_i^C = \begin{cases} \arg\min_{\langle \ell_i^C, u_j^C \rangle \in \mathcal{T}_i[\mathrm{k}_i]} \ell_i^C & \mathcal{T}_i[\mathrm{k}_i] \neq \emptyset \\ 1 & \text{otherwise} \end{cases} \quad (4)$$

that is, the greatest value of σ_i along the current $\boldsymbol{p}_i$ which can be achieved without colliding with other robots.

f) Status: To enforce coordination, we require each robot r_i to broadcast at each k_i its status $s_i[\mathrm{k}_i] = \langle \mathrm{ID}_i, \pi_i, \mathcal{E}_i[\mathrm{k}_i], \tilde{\Sigma}_i[\mathrm{k}_i], \tilde{\mu}_i^{\Delta}[\mathrm{k}_i] \rangle$. This contains a prediction of the robot's minimum-time trajectory traversed before the message is received, which consists in: 1) the robot chunk $\mathcal{E}_i[\mathrm{k}_i]$ with a proper δ_i^{h} defined according to Section III-D; 2) an estimated minimum-time temporal profile $\tilde{\Sigma}_i[\mathrm{k}_i] = \bigcup_{\sigma \in [L_i[\mathrm{k}_i], U_i[\mathrm{k}_i]]} \tilde{t}_i[\sigma]$, with $\tilde{t}_i[\sigma]$ being the estimated time to achieve the arc length σ along the current chunk $\mathcal{E}_i[\mathrm{k}_i]$[3]; 3) a proper future decision point $\tilde{\mu}_j^{\Delta}[\mathrm{k}_i]$, whose value is computed according to (2), with Δ defined as detailed in Section III-D.

[3]We use infinite values when robot r_i is parked along another robot's path (let $\ell_i^C = u_i^C = 0$ in such cases).

Algorithm 1: Each r_i's Coordination Algorithm (Run ning Each Eiscrete Time $\mathrm{k}_i \in \mathbb{N}$).

Input: robot state $(q_i, \dot{q}_i, \ddot{q}_i)$ at time $t_i = \mathrm{k}_i T_i^c$; set of neighbors' status $\bigcup_{r_j \in \mathcal{N}_i[\mathrm{k}_i]} s_j[\mathrm{k}_i]$; (possibly null) new posted goal $q_i^{\mathrm{g,new}}$.
Init ($\mathrm{k}_i = 0$)**:** $\alpha_i = \max_{r_j \in \mathcal{R}\setminus\{r_i\}} \lceil (T_j^c + \zeta)/T_i^c \rceil$; $\bar{\sigma}_i[-1] = 0$; $\tilde{\Sigma}_i[-\beta] = \{0\}$, $\beta \in [1, \alpha_i]$ (see Sec. III-D);
1 $\mathcal{C}_i[\mathrm{k}_i] \leftarrow \emptyset$; $\tilde{\Sigma}_i[\mathrm{k}_i] \leftarrow \emptyset$; $\bar{\sigma}_i[\mathrm{k}_i] = -1$;
2 **if** $q_i^{\mathrm{g,new}} \neq \emptyset$ **then**
3 compute the new path $\boldsymbol{p}_i$ from q_i to $q_i^{\mathrm{g,new}}$;
4 $\mathrm{k}_i \leftarrow 0$; $\bar{\sigma}_i[-1] \leftarrow 0$; $\tilde{\Sigma}_i[-\beta] \leftarrow \{0\}$;
5 update $\mathcal{E}_i[\mathrm{k}_i]$ (store $U_i[\mathrm{k}_i]$) and $\tilde{\mu}_i^{\Delta}[\mathrm{k}_i]$ with $\Delta = T_i^c$;
6 $\tilde{\mu}_i^{\Delta}[\mathrm{k}_i] \leftarrow \min\{\bar{\sigma}_i[\mathrm{k}_i - 1], \tilde{\mu}_i^{\Delta}[\mathrm{k}_i]\}$;
7 $s_i[\mathrm{k}_i] \leftarrow \langle \mathrm{ID}_i, \pi_i, \mathcal{E}_i[\mathrm{k}_i], \tilde{\Sigma}_i[\mathrm{k}_i - \beta], \tilde{\mu}_i^{\Delta}[\mathrm{k}_i] \rangle$;
8 update the set $\mathcal{C}_i[\mathrm{k}_i]$ of critical sections;
9 **if** $\mathrm{k}_i > 0$ **then**
10 **forall** $C \in \mathcal{C}_i[\mathrm{k}_i], \ell_i^C \leq U_i[\mathrm{k}_i - \alpha_i]$ **do**
11 $\langle \ell_h^C, u_k^C \rangle \leftarrow$ get precedence order according to Alg. 2 (using $s_i[\mathrm{k}_i]$ and $\{s_j[\mathrm{k}_i]\}_{r_j \in \mathcal{N}_i[\mathrm{k}_i]}$);
12 **if** $h = i \wedge \bar{\sigma}_i[\mathrm{k}_i] = -1$ **then** $\bar{\sigma}_i[\mathrm{k}_i] \leftarrow \ell_i^C$;
13 $\bar{\sigma}_i[\mathrm{k}_i] \leftarrow$ revise according to (4);
14 forward critical point $\bar{\sigma}_i[\mathrm{k}_i]$ to low-level controllers;
15 estimate (and store) min-time temporal profile $\tilde{\Sigma}_i[\mathrm{k}_i] = \bigcup_{\sigma \in [L_i[\mathrm{k}_i], U_i[\mathrm{k}_i]]} \tilde{t}_i(\sigma)$;
16 estimate future $\tilde{\mu}_j^{\Delta}[\mathrm{k}_i]$ with Δ def. according to Sec. III-D ;
17 broadcast status $s_i[\mathrm{k}_i] \leftarrow \langle \mathrm{ID}_i, \pi_i, \mathcal{E}_i[\mathrm{k}_i], \tilde{\Sigma}_i[\mathrm{k}_i], \tilde{\mu}_j^{\Delta}[\mathrm{k}_i] \rangle$;
18 sleep until control period T_i^c has elapsed;

B. The Coordination Algorithm

a) Main loop: Alg. 1 realizes each robot r_i's high-level control loop to compute, revise, and regulate access to critical sections along its path. At each iteration, the algorithm receives as input an updated robot state $(q_i, \dot{q}_i, \ddot{q}_i)$ and the last received status of neighboring robots, both sampled at each $t_i = \mathrm{k}_i T_i^c$, and newly assigned goals (if the robot is idle). The current chunk $\mathcal{E}_i[\mathrm{k}_i]$ is updated according to (1) (line 5). Also, the last feasible point the robot can be required to yield in the current period is revised according to (2) with $\Delta = T_i^c$ and to the value of the critical point at the previous iteration (lines 5–6). Then, the local set of critical sections $\mathcal{C}_i[\mathrm{k}_i]$ is computed by obtaining the pairwise intersections of $\mathcal{E}_i[\mathrm{k}_i]$ and the chunks $\{\mathcal{E}_j[\mathrm{k}_i]\}_{r_j \in \mathcal{N}_i[\mathrm{k}_i]}$ (line 8), respectively. These, along with the revised status (line 7) and the last ones received from neighboring robots, are used to revise the critical point (lines 10–12), which is then forwarded to the robot's low-level controller (line 14). The future minimum-time trajectory (line 15) and decision point (line 16) are then estimated via forward propagation of the robot kinodynamics. Finally, the new status message is broadcast (line 17).

b) Revise function: The core of the coordination algorithm is Alg. 2, which revises the precedence constraint regulating access to a critical section $C \in \mathcal{C}_i[\mathrm{k}_i]$ according r_i's own status and that of the other robot r_j involved in the critical section. For each $C \in \mathcal{C}_{ij}[\mathrm{k}_i]$, if either robot r_i cannot stop before the beginning of the critical section in the current $[\mathrm{k}_i]$ or robot r_j will not be able to revise its decision at time $[\mathrm{k}_j + 1]$ (with $[\mathrm{k}_i] \cap [\mathrm{k}_j]$), then the precedence constraint is set to let the constrained robot cross the section first (lines 7–8). However, there exist cases in which both such conditions do not hold (lines 1–6). As we will see in detail in Section III-D, this situation may happen if the current chunk ends in a critical section (line 1).

Algorithm 2: Revise Precedence Constraint.

Input: critical section $C \in \mathcal{C}_{ij}[\mathrm{k}_i]$; own robot status $s_i[\mathrm{k}_i]$; last received status $s_j[\mathrm{k}_i]$ of robot $r_j \in \mathcal{N}_i[\mathrm{k}_i]$.
Output: either $\langle \ell_i^C, u_j^C \rangle$, i.e., r_i yields for r_j at C, or $\langle \ell_j^C, u_i^C \rangle$ (vice versa).
1 $\mathrm{B}_h \leftarrow \tilde{\mu}_h^{\Delta}[\mathrm{k}_h] \geq \ell_h^C \wedge u_h^C > U_h[\mathrm{k}_h]$, $\forall h \in \{i, j\}$;
2 **if** $\mathrm{B}_i \wedge \mathrm{B}_j$ **then**
3 **throw** emergency brake and update $\bar{\sigma}_i[\mathrm{k}_i]$ accordingly;
4 **if** $\pi_i < \pi_j \vee (\pi_i = \pi_j \wedge \mathrm{ID}_i < \mathrm{ID}_j)$ **then**
5 start re-plan (see Sec. III-E);
6 **return** $\langle \ell_i^C, u_j^C \rangle$;
7 **if** $\tilde{\mu}_i^{\Delta}[\mathrm{k}_i] > \ell_i^C \vee \tilde{\mu}_j^{\Delta}[\mathrm{k}_i] > \ell_j^C$ **then**
8 **return** $\langle \ell_h^C, u_k^C \rangle$, $r_h, r_k \in \{r_i, r_j\} \mid \tilde{\mu}_h^{\Delta}[\mathrm{k}_i] > \ell_h^C$, $r_h \neq r_k$.
9 $\langle \ell_h^C, u_k^C \rangle \leftarrow$ compute according to heuristic h_{ij} (e.g., Alg. 3) using $s_i[\mathrm{k}_i]$ and $\{s_j[\mathrm{k}_i]\}_{r_j \in \mathcal{N}_i[\mathrm{k}_i]}$;
10 **return** $\langle \ell_h^C, u_k^C \rangle$

Algorithm 3: Prioritized MCT Heuristic.

Input: critical section $C \in \mathcal{C}_{ij}[\mathrm{k}_i]$; own robot status $s_i[\mathrm{k}_i]$; last received status $s_j[\mathrm{k}_i]$ of robot $r_j \in \mathcal{N}_i[\mathrm{k}_i]$.
Output: either $\langle \ell_i^C, u_j^C \rangle$, i.e., r_i yields for r_j at C, or $\langle \ell_j^C, u_i^C \rangle$ (vice versa).
1 **if** $\pi_i \neq \pi_j$ **then**
2 **return** $\langle \ell_h^C, u_k^C \rangle$, $r_h, r_k \in \{r_i, r_j\} \mid \pi_h < \pi_k, r_h \neq r_k$.
3 **if** $\tilde{t}_i(\ell_i^C) = \infty \vee \tilde{t}_j(\ell_j^C) = \infty$ **then**
4 **return** $\langle \ell_h^C, u_k^C \rangle$, $r_h, r_k \in \{r_i, r_j\} \mid [\mathrm{ID}_h < \mathrm{ID}_k, r_h \neq r_k$ *if* $\min\{\tilde{t}_i(\ell_i^C), \tilde{t}_j(\ell_j^C)\} = \infty] \vee [\tilde{t}_h(\ell_h^C) = \infty]$.
5 $\tau_i \leftarrow \tilde{t}_j(u_j^C) - \tilde{t}_i(\ell_i^C)$, $\tau_j \leftarrow \tilde{t}_i(u_i^C) - \tilde{t}_j(\ell_j^C)$;
6 **if** $\tau_h < \tau_k$ **then return** $\langle \ell_h^C, u_k^C \rangle, r_h, r_k \in \{r_i, r_j\}, r_h \neq r_k$;
7 **return** $\langle \ell_h^C, u_k^C \rangle$, $r_h, r_k \in \{r_i, r_j\} \mid \mathrm{ID}_h < \mathrm{ID}_k, r_h \neq r_k$.

In all other cases, both the robots can safely be required to stop before entering C. Thus, the precedence order is decided according to a heuristic function h_{ij} (lines 9–10).

In this paper, we use the prioritized version of the Minimum Completion Time (MCT) heuristic shown in Alg. 3. Specifically (see also Fig. 2c), robots are first prioritized according to their priority levels (lines 1–2). If this is not possible, the algorithm tries to minimize the time required by the yielding robot to exit the critical section (lines 5 and 6). Robot IDs are used to order robots if the previous criteria do not provide an ordering (lines 5 and 7).

c) Complexity: Similarly to the original centralized approach, Alg. 1 has a per-robot complexity which is *linear* in $|\mathcal{C}_i[\mathrm{k}_i]|$ (against the global complexity $|\mathcal{C}|$, with $\mathcal{C}$ being the set of all critical sections along the robots' paths $\{\boldsymbol{p}_i\}_{r_i \in \mathcal{R}}$ updated in a centralized manner all at once when a new mission is posted). Differently from [10], complexity is (upper-)bounded by a fixed quantity and thus not affected by the length of paths since $|\mathcal{C}_i[\mathrm{k}_i]| \leq \lceil \gamma_i \delta_i^{\mathrm{h}} / 2\mathrm{circ}(R_i) \rceil C_{m_i,2}$, with $C_{m_i,2} = m_i(m_i - 1)/2$ being the pairwise combinations of the maximum number of robots m_i that can be in the communication range $\mathcal{S}_i$.

d) Communication load: The distributed coordination based on Alg. 1 requires a communication bandwidth equal to $\sum_{r_i \in \mathcal{R}} |\mathcal{N}_i| \, \text{size\_of}\{s_i\}/T_i^c, |\mathcal{N}_i| \leq m_i$ (bit/sec) against the $N(\text{size\_of}\{s_i^C\}/T_i^c + \text{size\_of}\{\bar{\sigma}_i\}T_C^{-1})$ (bit/sec) of the original centralized approach, with s_i^C being the status message sent in a centralized manner and T_C being the period of the centralized fleet coordinator. The larger the communication radius, the larger $|\mathcal{N}_i|$ and m_i and therefore the larger the complexity and the load of the distributed framework.

C. Other Assumptions

Similarly to [5], the boundary conditions of our multi-robot distributed coordination framework are the following:

- A1) All robots are idle at time $t = 0$ and placed in a starting configuration such that $R_i(q_i(0)) \cap R_j(q_j(0)) = \emptyset$ for all pairs $(r_i, r_j) \in \mathcal{R}^2, i \neq j$.
- A2) Robots are not in motion when idle.
- A3) The low-level controllers of all robots ensure: a) bounded spatial errors such that $R_i(q_i(t)) \subseteq \mathcal{E}_i(t), \forall t$; b) the ability to stop in dynamically feasible critical points.
- A4) Robots do not back up along their paths.
- A5) Communication is *asynchronous*. In order to simplify the analysis below, we also assume that it is *ideal*, that is, messages are neither delayed nor lost (and therefore $\zeta = 0$). Note that, similarly to [5], the framework can be extended to handle bounded communication delays by setting ζ to be a conservative estimation of all transmission delays.

D. Considerations on Safety

Theorem 1 (Safety): Under the assumptions A1–A5, Alg. 1 with chunk horizons defined as

$$\delta_i^{\mathrm{h}} \triangleq \gamma_i^{-1}\{\Delta \bar{v}_i T_i^c + \bar{d}_i\}, \tag{5}$$

and communication radius

$$\rho \geq \rho_s, \; \rho_s \triangleq \max_{(r_i, r_j) \in \mathcal{R}^2} (\rho_i^s + \rho_j^s),$$

$$\rho_i^s \triangleq \mathrm{circ}(R_i) + \gamma_i \delta_i^{\mathrm{h}} \quad \forall r_i \in \mathcal{R}, \tag{6}$$

ensures that robots do not collide with each other if: a) $\Delta \geq (1 + \alpha_i)T_i^c$ and precedence constraints are decided according to shared heuristic and static information; b) $\Delta \geq (2 + \alpha_i)T_i^c$, with α_i defined according to (3).

Proof: Collision may happen whenever there exists a pair $(\sigma_i(t), \sigma_j(t))$ such that $R_i(\boldsymbol{p}_i(\sigma_i(t)) \cap R_j(\boldsymbol{p}_j(\sigma_j(t)) \neq \emptyset$. Under A1–A5, the centralized approach is proved to be safe [5]. In a distributed framework, this property may not hold due to asynchronous communication, data locality and/or loss of agreement, each of which may cause wrong computations of critical sections or of precedence constraints. Henceforth, let the symbols $(\hat{\sigma}_i, \hat{\sigma}_j)$ refer to a pair of colliding configurations. Due to A3.a, if a collision happens, then $\mathcal{E}_i(t) \cap \mathcal{E}_j(t) \neq \emptyset$.

[Feasibility] A3.b and Alg. 2 with each robot's own decision point $\tilde{\mu}_i^{\Delta}[\mathrm{k}_i]$ computed according to lines 5–6 of Alg. 1 ensure feasibility, that is, robots can yield in their critical points if required to.

[Communication radius] If both (5) and (6) hold, then $\mathcal{E}_i(t) \cap \mathcal{E}_j(t) = \emptyset$ for each robot r_j entering a robot r_i's communication range (since $\rho_i^s \geq \mathrm{circ}(R_i) + \mathrm{circ}(\mathcal{E}_i)$, see Fig. 2a). Also, both the robots may safely communicate to each other their current chunk without colliding only if $\Delta \geq (1 + \alpha_i)T_i^c$ (it can be proved via worst-case analysis). Note that the addition of $\mathrm{circ}(R_i)$ and $\mathrm{circ}(R_j)$ ensures avoiding collisions while entering into the communication range, i.e., for all times t such that $\mathrm{circ}(R_j(q_j(t))) \cap \mathcal{S}_i(q_i(t), t) \wedge W_j(q_j(t)) \notin \mathcal{S}_i(q_i(t), t)$.

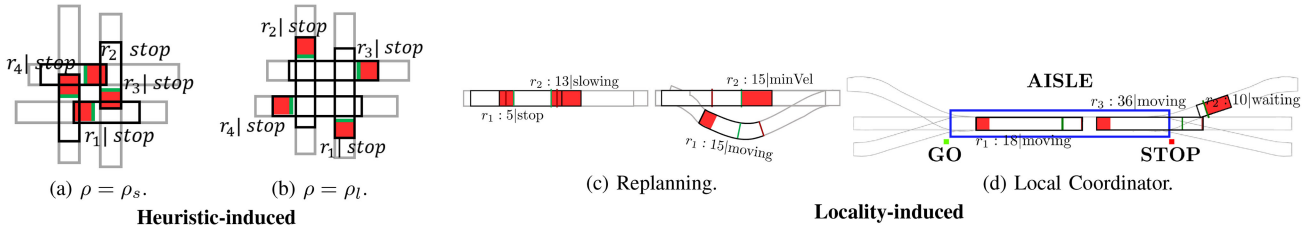
[Asynchronous communication - 1] Each r_i's local view at time k_i consists in its status $s_i[\mathrm{k}_i]$ and the last ones received by its neighbors $\{s_j[\mathrm{k}_i]\}_{r_j \in \mathcal{N}_i[\mathrm{k}_i]}$, with $s_j[\mathrm{k}_i] \in \{s_j[\mathrm{k}_j - \beta]\}_{\beta=0}^{\alpha_j}$. Each received $s_j[\mathrm{k}_i]$ includes r_j's estimated decision point $\tilde{\mu}_j^{\Delta}[\mathrm{k}_i] \in \{\tilde{\mu}_j^{\Delta}[\mathrm{k}_j - \beta]\}_{\beta=0}^{\alpha_j}$. Therefore, each r_i can know the last point its neighbors are able to yield in given their current k_j even when $\beta = \alpha_j$, only if $\Delta \geq (1 + \alpha_i)T_i^c$.

[Asynchronous communication - 2] Note that for each $[\mathrm{k}_i] \cap [\mathrm{k}_j]$, each robot r_i views all critical sections $C \in \mathcal{E}_i[\mathrm{k}_i] \cap \mathcal{E}_j[\mathrm{k}_j - \beta_j], \beta_j \in [0, \alpha_j]$. Therefore, r_i may not see the critical sections $C \in \mathcal{E}_i[\mathrm{k}_i] \cap \mathcal{E}_j[\mathrm{k}_j] \setminus \mathcal{E}_j[\mathrm{k}_j - \beta_j]$, and hence possibly collide with r_j, as exemplified in Fig. 2d. The conditions $\bar{\sigma}_i[\mathrm{k}_i] \leq U_i[\mathrm{k}_i - \alpha_i]$ (forced by (4)) and (3), A3 and A4 (which implies $U_i[\mathrm{k}_i] \leq U_i[\mathrm{k}_i + 1]$ for all k_i), is then necessary to ensure robot r_i can reach $\hat{\sigma}_i$ only when all robots have received a chunk containing $\hat{\sigma}_i$. As a consequence of constraining the upper bound critical points, then $\hat{\sigma}_i \in [L_i[\mathrm{k}_i], U_i[\mathrm{k}_i - \alpha_i]]$ and $\hat{\sigma}_j \in [L_j[\mathrm{k}_j], U_j[\mathrm{k}_j - \alpha_j]]$.

[Correctness (with agreement)] If $\Delta = (1 + \alpha_i)T_i^c$, then each $r_j \in \mathcal{N}_i[\mathrm{k}_i]$ can safely stop at all $\sigma_j \geq \tilde{\mu}_j^{\Delta}[\mathrm{k}_i]$ in its current $[\mathrm{k}_i]$. For simplicity, let us assume both robots see a common C at times $[\mathrm{k}_i]$, $[\mathrm{k}_j]$, $[\mathrm{k}_i] \cap [\mathrm{k}_j]$, i.e., $\ell_i^C, u_i^C \in \mathcal{E}_i[\mathrm{k}_i] \cap \mathcal{E}_i[\mathrm{k}_i - \beta_i]$ and $\ell_j^C, u_j^C \in \mathcal{E}_j[\mathrm{k}_j] \cap \mathcal{E}_i[\mathrm{k}_j - \beta_j]$. With reference to Fig. 2b, robot r_j at time $[\mathrm{k}_j]$ knows if r_i can yield at ℓ_i^C in $[\mathrm{k}_i + \alpha_i]$, while robot r_i knows if r_j can yield at C in $[\mathrm{k}_j]$. Assume both can yield; if both robots decide to enter C (disagreement) and drive with maximum acceleration, then a collision may happen. Conversely, if only one robot $r_h \in \{r_i, r_j\}$ decides to enter C (agreement) the other will safely yield in $\ell_h^C[\mathrm{k}_h]$. Agreement can be forced by using shared totally ordering heuristics based on static information.

[Correctness (with disagreement)] Safety can be preserved in case of disagreement while allowing at least one robot to be informed and then revise its decision over a critical section according to the other robot behavior. This is achieved by choosing $\Delta \geq (2 + \alpha_i)T_i^c$: with reference of Fig. 2b, $\tilde{\mu}_j^{\Delta}[\mathrm{k}_i] \leq \ell_i^C$ and $\tilde{\mu}_i^{\Delta}[\mathrm{k}_j] \leq \ell_j^C$ and both the robots decide at $[\mathrm{k}_i]$ and $[\mathrm{k}_j]$ to enter C, then robot r_i can be informed on time and revise its decision at time $[\mathrm{k}_i + \alpha_i]$. Note that, if $\Delta \geq (2 + \alpha_i)T_i^c$ allows to decouple safety by agreement, disagreement may still affect performance, causing deadlocking situations in which both robots wait for each other, see Section III-E). Specifically, the more selfish the two robots, the lower the probability that deadlocks will happen. As shown in the next point, Alg. 3 seeks to minimize such situations.

[Agreement] If each robot r_i takes decisions at each k_i according to $\tilde{\Sigma}_i[\mathrm{k}_i - \alpha_i]$ and $\tilde{\Sigma}_i[\mathrm{k}_i + 1]$ can only be delayed w.r.t. $\tilde{\Sigma}_i[\mathrm{k}_i]$ (i.e., $\tilde{\Sigma}_i[\mathrm{k}_i]$ is an optimistic realization of the real trajectory of the robot), then Alg. 3 may not ensure agreement.



(a) $\rho = \rho_s$. (b) $\rho = \rho_l$. (c) Replanning. (d) Local Coordinator.

Fig. 3. Deadlock situations and strategies to prevent (a)-(b) or recover (c)-(d) from them.

For simplicity, assume both robots see the same C. Each r_i takes decisions according to $\tilde{\Sigma}_i[\mathrm{k}_i - \alpha_i]$ and $\tilde{\Sigma}_j[\mathrm{k}_i] \in \{\tilde{\Sigma}_i[\mathrm{k}_j - \beta]\}_{\beta=0}^{\alpha_j}$. A conflict may arise when both the robots decides to have priority over C according to their local view, which may happen only if there exists four slack variables $\epsilon_k \geq 0$ such that both the following equations hold:

$$\tau_i|_{\mathrm{k}_i-\alpha_i,\mathrm{k}_j-\alpha_j} + \epsilon_1 > \tau_j|_{\mathrm{k}_i-\alpha_i,\mathrm{k}_j-\alpha_j} - \epsilon_2$$

$$\tau_i|_{\mathrm{k}_i-\alpha_i,\mathrm{k}_j-\alpha_j} - \epsilon_3 < \tau_j|_{\mathrm{k}_i-\alpha_i,\mathrm{k}_j-\alpha_j} + \epsilon_4$$

which is always verified since $\epsilon_1 > -\epsilon_2 - \epsilon_3 - \epsilon_4$. Therefore, both the robots will try to minimize their completion times. Selfishness decreases with a more recent (hence less optimistic) trajectory $\tilde{\Sigma}_i[\mathrm{k}_i - \beta]$, $\beta \in [1, \alpha_i]$, in line 7 of Alg. 2. Note that different views of C may affect the estimations of the times $\tilde{t}_i(\ell_i^C)$ and $\tilde{t}_j(\ell_j^C)$. However, since robots may only progress along their paths (A4), this may only slightly reduce selfishness.

[Clock de-synchronization] Note that, under the assumption that at least one status message $s_j[\mathrm{k}_i] \in \{s_j[\mathrm{k}_j - \beta]\}_{r_j \in \mathcal{N}_i[\mathrm{k}_i]}, \beta \in [0, \alpha_j]$ is received from all r_i's neighbors at each time (A5), clock de-synchronization can only affect efficiency (i.e., τ_i and τ_j are affected by errors), not safety.

E. Considerations on Liveness

Assuming well-formed goals, data locality may cause loss of liveness only due to deadlocks [4], i.e., cyclic precedence assignments among robots over sets of critical sections. Two types of deadlocks may occur: heuristic-induced (lines 9–10 of Alg. 2) and locality-induced (lines 2–6 of Alg. 2).

a) Heuristic-induced: These deadlocks are caused by disagreement on a precedence order among two robots or by the absence of a total order on pairwise overlapping configurations. Filtering precedence orders to enforce kinematic feasibility (lines 7–8 of Alg. 2) may lead to the absence of a total order among robots sharing overlapping critical sections, i.e., such that the distance between two critical sections in $\mathcal{C}_i[\mathrm{k}_i]$ is less than one footprint (see Fig. 3a). This loss of liveness may happen also while using totally ordering heuristics [4] such as ordering robots according to their IDs. We prevent these deadlocks by using a communication radius $\rho_\ell \triangleq \max_{(r_i,r_j)\in\mathcal{R}^2}(\rho_i^s + \mathrm{circ}(R_i) + \rho_j^s + \mathrm{circ}(R_j)) > \rho_s$ and a chunk horizon $\delta_i^{\mathrm{h}} = \gamma_i^{-1}\{\Delta\bar{v}_i T_i^c + \bar{d}_i + \mathrm{circ}(R_i)\}$. As a consequence, each robot r_i can always 1) recognize and 2) therefore stop before accessing chains of overlapping critical sections in $\mathcal{C}_i[\mathrm{k}_i]$ such that r_i has precedence in some but not all of them (see Fig. 3b). Then, any totally ordering heuristic may be used to unlock the deadlocked situation.

b) Locality-induced: These deadlocks may happen when two robots access a shared region of the environment before being aware of each other due to range-limited communication. Theorem 1 ensures safety even in this situation (under a proper choice of the chunk horizon and of the communication radius). However, both robots would stop to avoid the collision, hence potentially leading to deadlock (line 3 of Alg. 2).

Deadlocks may be globally prevented at exponential cost by leveraging distributed constraint-based methods and infinite horizons [20]. Aiming at large fleets, two strategies (prevention and recovery) are shown in Fig. 3c-3d. The first (see Fig. 3c) exploits (prioritized) replanning as proposed in [4]: the robot with lower priority replans its path toward the goal while considering the other robots in its neighborhood as obstacles (line 5 of Alg. 2).

Replanning might fail when robots navigate in narrow environments, such as warehouse aisles (see Fig. 3 d). To overcome the shortcomings caused by range-limited communication, as in [13] and [21], we exploit a hierarchical solution based on local coordinators responsible for scheduling access to each narrow space. As a robot r_i communicates a decision point $\tilde{\mu}_i^{\Delta}$ inside the narrow space, the local coordinator shares this information with any vehicle r_j approaching in the opposite direction, and establishes temporary one-way traffic, via a precedence constraint on r_j. An example is shown in Fig. 3d, where access regulation is depicted by traffic lights. Other strategies, e.g., based on multi-path evaluation [9] or on traffic-flow [22], may be exploited in future work bias motion planning so as to avoid head-to-head conflicts.

IV. Evaluation

We validate our approach via simulations while focusing on two main points: performance in terms of safety and liveness (Test 1), and the gain in scalability while comparing the proposed approach against the centralized coordinator of [10] (Test 2). Selected moments of all simulations are shown in the video available at https://www.youtube.com/watch?v=EUtQvMW60Uc.

a) Setup: Paths were computed using sampling-based motion planners (RRTConnect [23]). The conservative model g_i is based on a trapezoidal velocity profile with maximum velocity $\bar{v}_i$ and constant acceleration/deceleration $\underline{a}_i$. The prioritized version of the Minimum Completion Time (MCT) heuristic (and hence $\Delta = (2 + \alpha_i)T_i^c$) is used for all the tests, and narrow spaces are handled by the local coordinator structure presented in Section III-E. All the tests ran on an Intel Core i5-5250 U dual-core 1.60 GHz processor with 8 GB 1600 MHz of DDR3 RAM. Alg. 1 is implemented in Java and available as open source[4].

b) Test 1 – Safety and liveness: This test was performed in a simulated environment consisting of several rooms and tight spaces (Fig. 4). We used a heterogeneous fleet of nine vehicles, whose parameters are listed in Table I. Four different values of

[4]https://github.com/miche-sr/DistributedFleets

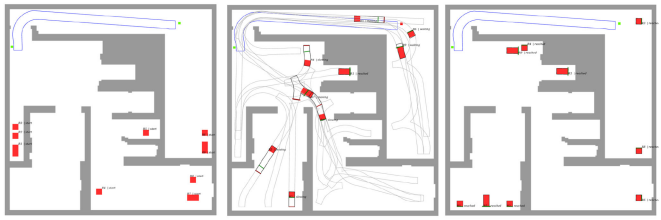


Fig. 4. Test 1: Nine robots in a realistic environment traveling from initial poses to assigned goals (from left to right).

TABLE I
TEST 1: SAFETY AND LIVENESS

Radius	Completed Tests	Safe Tests	Safe Critical Sections
$\rho < \rho_s$	40 %	80 %	99 %
$\rho = \rho_s$	40 %	100 %	100 %
$\rho = \rho_\ell$	90 %	100 %	100 %
$\rho > \rho_\ell$	80 %	100 %	100 %

Robot parameters: r_1, r_7, and r_9: $(\bar{v}_i, \underline{a}_i) = (\pm 4.0\,\text{m/s}, \pm 2.0\,\text{m/s}^2)$ and $T_i^c = 150\,\text{ms}$. Other robots: $(\bar{v}_i, \underline{a}_i) = (\pm 3.0\,\text{m/s}, \pm 1.0\,\text{m/s}^2)$ and $T_i^c = 350\,\text{ms}$. Column 3: $(\#\text{sim.})^{-1} \sum_{\text{sim.}} |\{C \in \mathcal{C} : \text{safe}\}|/|\mathcal{C}|$, with $\mathcal{C}$ being the set of all critical sections computed in a centralized manner as in [10], [5].

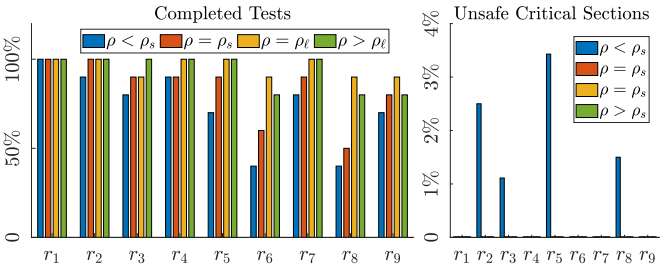


Fig. 5. Test 1: Liveness and safety indices per robot with different communication radii. Unsafe critical sections for each robot r_i are computed as $(\#\text{sim.})^{-1} \sum_{\text{sim.}} |\{C \in \mathcal{C}_i : \text{unsafe}\}|/|\mathcal{C}_i|$.

the communication radius have been tested: R1) $\rho < \rho_s$, where we expect loss of safety; R2) $\rho = \rho_s$ to ensure safety, computed as in (6); R3) $\rho = \rho_\ell$ to avoid heuristic-induced deadlocks; R4) $\rho > \rho_\ell$ where more information is shared. For each case, we performed 10 simulations to obtain statistically meaningful results. The initial and final configurations are fixed, but the robots follow different paths in every simulation. Tests are completed only if there are no collisions and deadlocks. Results, displayed in Table I, show: 1) the percentage of competed and safe tests for each case R1–R4 (column 2 and 3); 2) the average percentage of critical sections traversed without collisions among the 10 simulations (column 4). Statistics about each robot are shown in Fig. 5 and report the number of completed tests and the average percentage of the critical sections involving a collision.

c) Discussion: Results shown in Table I confirm the theoretical claim in Theorem 1 of provable safety for all cases in which $\rho \geq \rho_s$ and empirically demonstrate that $\rho \geq \rho_l$ prevents heuristic-induced deadlocks (see our considerations on liveness in Section III-E). An interesting outcome is the reduction of total completed tests for R4) w.r.t. R3): chunk horizons longer than ρ_l alone are useless to prevent heuristic-induced deadlocks and may not be effective to prevent locality-induced deadlocks. Specifically, all safe yet not completed tests in Fig. 5 when $\rho \geq \rho_\ell$

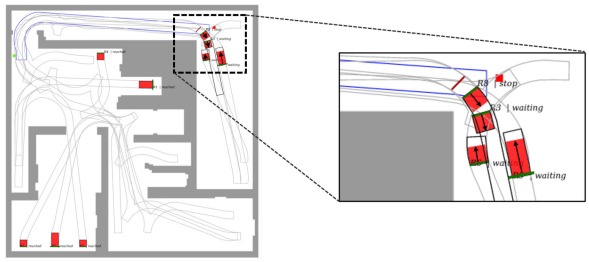


Fig. 6. Test 1: Example of an undesired locality-induced deadlock.

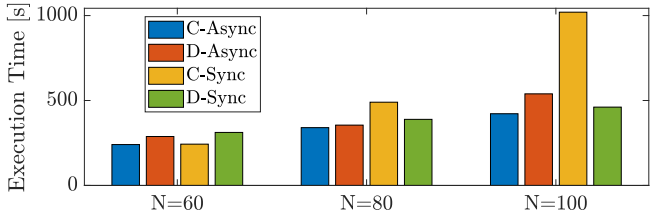


Fig. 7. Test 2: Total execution time with different number of robots, different coordination algorithms, and different goal assignment strategies.

were not completed due to the occurrence of locality-induced deadlocks provoked by multiple robots waiting to access the narrow space (see Fig. 6). However, this loss of liveness, caused by a simplistic implementation of the access policy, might be overcome by exploiting dedicated parking spots where robots can wait without creating conflicts.

d) Test 2 – Scalability: We set up a simulation where N homogeneous robots are equally spaced along a circle and travel along its diameter. This scenario causes all paths to intersect in a single choke-point in the center and therefore shows the limitations of the centralized approach in terms of complexity. The overhead of computing the global set of critical sections causes the centralized coordinator to exceed its control period T_C and hence safety guarantees may be lost. Increasing T_C to regain safety reduces reactivity, thus leading to low performance. Conversely, complexity is fixed in the distributed approach and may be reduced due to the actual cardinality of $\mathcal{N}_i$ (see Section III-B c). We present four scenarios: S1) **C-Async**: goals are periodically assigned to idle robots (every 2 secs) and coordination is centralized; S2) **D-Async**: goal scheduling is the same as in C-Async, but the robots are coordinated with Algorithm 1; S3) **C-Sync**: goals to robots are synchronously assigned and coordination is centralized; S4) **D-Sync**: goal scheduling as in S3 but coordination is distributed. S1–S4 have been tested with $N = \{60, 80, 100\}$ robots. The control period T_C used by the centralized coordinator is 2000 ms, whereas, in the distributed approach each robot has a control period $T_i^c = 350$ ms. In each test, we analyze: 1) the time to complete the simulation, 2) the actual control period $\tilde{T}_C$, 3) the time to compute the critical sections t_{CS} and 4) to update dependencies t_{dep}. The total execution times of each test S1–S4 are shown in Fig. 7. Tables II and IV report the results in terms of maximum and average $\tilde{t}_C$, t_{CS}, and t_{dep} (normalized w.r.t. the given control period T_C), together with an evaluation of the number of cycles in which the nominal control period T_C was violated. Note that we do not report max and mean values in C-Sync since all the critical sections are computed at once in the first control cycle. The

TABLE II
TEST 2: C-ASYNC

N	$\tilde{T}_C/T_C$		$\%\frac{\tilde{T}_C}{T_C} > 1$	t_{CS}/T_C		t_{dep}/T_C	
	max	mean		max	mean	max	mean
60	**1.30**	1.00	**11.7%**	0.06	0.03	0.17	0.06
80	**1.26**	**1.01**	**11.7%**	0.08	0.03	0.19	0.08
100	**3.35**	**1.15**	**11.2%**	0.07	0.03	0.31	0.14

TABLE III
TEST 2: D-ASYNC

N	$\tilde{T}_C/T_C$		$\%\frac{\tilde{T}_C}{T_C} > 1$	t_{CS}/t_C		t_{dep}/t_C	
	$\max_{\mathcal{R}}$	mean		$\max_{\mathcal{R}}$	mean	$\max_{\mathcal{R}}$	mean
60	1.0	1.0	0%	0.09	0.04	0.01	0.01
80	1.0	1.0	0%	0.29	0.01	0.01	0.003
100	1.0	1.0	0%	0.09	0.01	0.003	0.003

TABLE IV
TEST 2: C-SYNC

N	$\tilde{T}_C/T_C$		$\%\frac{\tilde{T}_C}{T_C} > 1$	t_{CS}/T_C	t_{dep}/T_C	
	max	mean			max	mean
60	**3.85**	**1.05**	**12.2%**	**1.07**	0.42	0.13
80	**25.5**	**2.1**	**11.9%**	**2.89**	**1.26**	0.27
100	**28**	**2.05**	**8.7%**	**3.94**	**2.89**	0.56

TABLE V
TEST 2: D-SYNC

N	$\tilde{T}_C/T_C$		$\%\tilde{T}_C/T_C > 1$		t_{CS}/T_C		t_{dep}/T_C	
	$\max_{\mathcal{R}}$	mean	$\max_{\mathcal{R}}$	mean	$\max_{\mathcal{R}}$	mean	$\max_{\mathcal{R}}$	mean
60	**1.1**	1.0	0.37%	0.01%	**1.1**	0.005	0.20	0.005
80	**1.7**	1.0	**1.14%**	0.19%	**1.6**	0.005	0.16	0.005
100	**1.1**	1.0	0.36%	0.01%	0.6	0.005	0.16	0.005

same indexes are displayed in Table III and V which summarize the results of D-Async and D-Sync, respectively. Mean values are computed as the mean of the average among robots, while maximum values consider the worst case among robots.

e) Discussion: Fig. 7 shows that, as expected, the centralized coordinator scales badly in this scenario, especially in C-Sync. Conversely, the growth is approximately linear in the distributed approach: with 100 robots, the time for D-Sync is less than half of C-Sync. On the other hand, when goals are posted asynchronously, only a subset of the worst-case critical sections are active at the same time.

Thus, C-Async and D-Sync have similar behaviors in terms of total times, with C-Async performing slightly better with $N = 100$. Also, $\tilde{T}_C > T_C$ for many cycles in both C-Sync and C-Async. Conversely, in the distributed cases, the robots never exceed their nominal T_i^c (case D-Async), or at least only on a negligible fraction of the control cycles.

V. CONCLUSION

We have presented a distributed method for coordinating heterogeneous fleets of autonomous robots with second-order dynamics. We formally prove that the method is safe assuming reliable communication, conservative (yet private) kinodynamic models, and a proper sizing of the communication radius. We also demonstrate empirically that the approach scales better than the centralized method from which it is derived. Future work will focus on the development of distributed deadlock identification and prevention/repair strategies, traffic-aware motion planning, and on testing with real robots.

REFERENCES

[1] N. Boysen *et al.*, "Warehousing in the e-commerce era: A survey," *Eur. J. Oper. Res.*, vol. 277, no. 2, pp. 396–411, 2019.

[2] G. Fragapane *et al.*, "Increasing flexibility and productivity in industry 4.0 production networks with autonomous mobile robots and smart intralogistics," *Ann. Oper. Res.*, Jul. 2020, doi: 10.1007/s10479-020-03526-7.

[3] Z. Yan *et al.*, "A survey and analysis of multi-robot coordination," *Int. J. Adv. Robot.*, vol. 10, no. 12, p. 399, 2013.

[4] A. Mannucci *et al.*, "On provably safe and live multirobot coordination with online goal posting," *IEEE Trans. Robot.*, pp. 1–19, 2021, doi: 10.1109/TRO.2021.3075371.

[5] "Provably safe multi-robot coordination with unreliable communication," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3232–3239, Oct. 2019.

[6] K.-C. Chen *et al.*, "Wireless networked multirobot systems in smart factories," in *Proc. IEEE*, vol. 109, no. 4, pp. 468–494, 2021.

[7] A. Ghaffarkhah *et al.*, "Communication-aware motion planning in mobile networks," *IEEE Trans. Autom. Control*, vol. 56, no. 10, pp. 2478–2485, Oct. 2011.

[8] S. Caccamo *et al.*, "Rcamp: A resilient communication-aware motion planner for mobile robots with autonomous repair of wireless connectivity," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2017, pp. 2010–2017.

[9] K. E. Bekris *et al.*, "Safe distributed motion coordination for second-order systems with different planning cycles," *Int. J. Robot. Res.*, vol. 31, no. 2, pp. 129–150, 2012.

[10] F. Pecora *et al.*, "A loosely-coupled approach for multi-robot coordination, motion planning and control," in *Proc. ICAPS*, vol. 28, no. 1, pp. 485–493, 2018.

[11] S. Schemmer *et al.*, "Reliable real-time cooperation of mobile autonomous systems," in *Proc. 20th Symp. Reliable Distrib. Syst. IEEE*, 2001, pp. 238–246.

[12] L. Cancemi *et al.*, "Distributed multi-level motion planning for autonomous vehicles in large scale industrial environments," in *Proc. IEEE 18th Conf. ETFA.*, 2013, pp. 1–8.

[13] I. Draganjac *et al.*, "Decentralized control of multi-agv systems in autonomous warehousing applications," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 4, pp. 1433–1447, Oct. 2016.

[14] G. R. de Campos *et al.*, "Traffic coordination at road intersections: Autonomous decision-making algorithms using model-based heuristics," *IEEE Intell. Transp. Syst. Mag.*, vol. 9, no. 1, pp. 8–21, Jan. 2017.

[15] P. Yu *et al.*, "A fully distributed motion coordination strategy for multi-robot systems with local information," in *Proc. Amer. Control Conf.*IEEE, 2020, pp. 1423–1428.

[16] P.-L. Götvall, "Volvo group collaborative robot systems laboratory: A collaborative way for academia and industry to be at the forefront of artificial intelligence," KI-Künstliche Intelligenz, vol. 33, no. 4, pp. 417–421, 2019.

[17] M. Čáp *et al.*, "Complete decentralized method for on-line multi-robot trajectory planning in well-formed infrastructures," in *Proc. ICAPS*, vol. 25, no. 1, pp. 324–332, 2015.

[18] H. Andreasson *et al.*, "Fast, continuous state path smoothing to improve navigation accuracy," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 662–669.

[19] L. Pallottino *et al.*, "Decentralized cooperative policy for conflict resolution in multivehicle systems," *IEEE Trans. Robot.*, vol. 23, no. 6, pp. 1170–1183, Dec. 2007.

[20] B. Faltings, "Distributed constraint programming," in *Proc. Foundations Artif. Intell.*. Elsevier, vol. 2, 2006, pp. 699–729.

[21] A. Palleschi *et al.*, "Toward distributed solutions for heterogeneous fleet coordination," in *Proc. 2nd IRIM Conf.*, 2020, pp. 1–3.

[22] L. Palmieri *et al.*, "Kinodynamic motion planning on gaussian mixture fields," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 6176–6181.

[23] J. J. Kuffner *et al.*, "Rrt-connect: An efficient approach to single-query path planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2000, pp. 995–1001.