Dynamic Distributed Coordination Schemes for Multi-Mobile Robot Systems under Collision Avoidance Constraints

Alessandro Casavola, Vincenzo D'Angelo, Ayman El Qemmah, Gianfranco Gagliardi, Francesco Tedesco, Franco Angelo Torchiaro

***Abstract*—Guaranteeing collision avoidance is of paramount importance in view of accomplishing missions where many agents are involved to share the same space. The main objective of this work is to expand the Turn-Based Command Governor approach in ([21]) by taking non-convex *Collision Avoidance* constraints into account when performing Plug-and-Play (PnP) operations among agents operating in a 2D environment. To deal with such a scenario, formal conditions that guarantee collision-free Plug-and-Play (PnP) operations are given. These conditions are based on the concept of safe areas, which define regions where agents can safely perform PnP operations without the risk of collision. The effectiveness of the proposed strategy is illustrated through various examples, highlighting its potential for mission accomplishment involving multiple agents.**

***Note to Practitioners*—This paper presents a novel distributed control strategy designed to coordinate multiple mobile agents safely and efficiently. By incorporating advanced control techniques such as Model Predictive Control and Command Governors, the proposed approach addresses the challenges of collision avoidance and dynamic formation changes. The methodology offers several potential benefits for practical applications, including enhanced safety, increased flexibility through Plug-and-Play capabilities, and reduced computational demands compared to centralized alternatives. Although implementation may require specialized expertise in control engineering, the potential impact of this research on robotics, autonomous vehicles, and other multi-agent systems is significant. This work provides a foundation for future research and development in distributed control and coordination.**

***Index Terms*—Collision Avoidance, Plug & Play, Distributed Command Governor, Multi-Mobile Robot Systems.**

I. Introduction

THE increasing prevalence of multi-agent systems demands robust solutions for safe and efficient operation, particularly in confined spaces. Traditional control methods often struggle with real-world constraints and dynamic formations, leading to potential collisions and inefficiencies. This paper addresses these challenges by developing a distributed predictive control scheme that explicitly incorporates collision avoidance and handles dynamic agent interactions.

A. Literature Review

All the authors are with DIMES Engineering Department, University of Calabria, Rende, Cosenza 87100 ITA (corresponding email: ftedesco@dimes.unical.it)

Significant theoretical and practical research advances towards distributed control of multiple mobile agent systems have been achieved during the last decades . This is mostly due to their potential applications in the real world and to specific advantages over traditional single-robot systems, such as lower cost and increased efficiency. In this respect, research efforts have been directed towards formation control of various robotic platforms, including microrobotic swarms [11], multiple manipulators, mobile robots [2], Autonomous Underwater Vehicles (AUVs) [3], and Unmanned Aerial Vehicles (UAVs) [7]. A crucial challenge in this domain is achieving coordinated behavior under constraints such as limited communication bandwidth, the necessity for collision avoidance (CA), and the complexities of dynamic environments. Several control strategies are employed to address these challenges. Genetic Algorithm based Linear-Quadratic Regulator is utilized in [11]. Consensus-based algorithms are prominent in distributed formation control [1, 2], enabling coordination through local information exchange [5, 6, 7, 10]. Other approaches include behavioral forces, virtual structures, and leader-follower methods, potential fields [2, 3, 4, 7, 8, 9].

Theoretical investigations often focus on the stability of multi-agent systems under different conditions, including communication delays, [2] while the importance of experimental validation is consistently highlighted, with several studies detailing real-world implementations of proposed control algorithms [1, 2, 3, 4]. Applications of these technologies span diverse fields such as intelligent transportaion [15], underwater exploration [3], and industrial automation [13, 2, 11].

Further methods using trajectory optimization [15, 11] are commonly exploited to generate collision-free trajectories, using sophisticated techniques such as path planning, decision trees, graph technique, heuristic search technique (please refer to [16] for a detailed summary). However, these methods are not able to take into account input constraints (actuator saturations) or state constraints (limited speed of the vehicles) and suffer significant replanning cycles when formation prescriptions are introduced to perform more complicated tasks.

B. Paper Contribution

This paper focuses on a distributed predictive control scheme in order to explicitly code input and/or state-related constraints. In this class of method the well known Model Predictive Control (MPC) is the most used and has been widely

investigated in tracking and formation problem of multi-agent system [17, 18]. In particular, this work considers the ideas behind multi-agent Command Governor (CG) for networked decoupled dynamical systems [19] and attempts to include *Collision Avoidance* constraints in the supervision problem at hand. As for the MPC schemes, CG-based approaches offer a more systematic, anticipatory, and scalable strategy for addressing distributed collision avoidance in multi-agent systems when contrasted with approaches such as barrier functions, potential fields, and velocity obstacles. Unlike the works cited in the preceding section, these methods enable explicit management of constraints, improve trajectory optimization, and exhibit enhanced performance in complex and dynamic environments, thereby providing direct avenues towards scalable methodologies. More in detail, potential field methods often suffer from local minima issues, which can lead to deadlocks or suboptimal paths, especially in complex dynamic environments. Similarly, velocity obstacle-based approaches require continuous online replanning, which increases computational overhead, particularly for large-scale multi-agent systems. Compared to potential fields, which may lead to local minima, our approach ensures constraint satisfaction through a structured turn-based mechanism, avoiding the pitfalls of local minima. Unlike velocity obstacle-based methods, which require frequent inter-agent updates, the proposed strategy selectively activates collision avoidance constraints only when necessary, thereby reducing communication and computational demands.

On the other hand, CG-based methods present also advantages when compared with MPC-based approaches. In fact, because additional constraints/conditions are often required to achieve closed-loop stability, such MPC-based approaches may be unsuitable for real-world applications. A further motivation for employing the CG is the frequent need for controller updates and redesign in traditional formation control methods to accommodate varying formation tasks [6]. This requirement often conflicts with the desire to maintain existing controller performance. By preserving the original controller, the CG offers a practical two-layer solution to establish required formation control strategies without controller modifications. As a consequence, an additional advantage of CG-based schemes compared to general MPC-based control methods is that they generally solve a simpler optimization problem online, which is less computationally intensive.

The proposed CG scheme follows the same lines of [20] where an early CG-based scheme for dealing with *Collision Avoidance* was presented. In particular, those constraints were recast in a mixed-integer format and embedded in an optimization problem that minimizes the gap between the desired set-point and a modified command admissible in view of the prescribed constraints.

However, that solution was only suitable for small static formations as the *Collision Avoidance* prescriptions were imposed in an all-to-all constraints topology so that even agents that are spatially well separated have to account for possible collisions in the optimization problem, resulting in a significant conservative behavior. Furthermore, as frequent information interchange is not desirable, such constraints should be imposed only when strictly necessary.

For this reason, that scheme has been here extended along two main directions: i) the concept of *turn*, introduced in [19], is here used to define collision avoidance constraints only for agents whose distance is lower than a certain quantity; ii) in order to deal with dynamic formations, *P*lug-a*n*d-*P*lay (PnP) operations among the agents are taken into account in order to enhance the overall system performance. In this respect, results from [21] are exploited in order to enhance the degree of modularity and flexibility of the solution, by achieving the capability of adding/removing subsystems without violating operative or coordination constraints, after the necessary reconfiguration of the control and supervision strategies.

A significant challenge arises due to the necessity of incorporating non-convex *Collision Avoidance* constraints, considering that in [21] only constraints of convex nature are taken into account. Consequently, this paper introduces conditions that guarantee to safely include constraints related to *Collision Avoidance* prescription.

A further contribution relies on a heuristic procedure, based on a token sharing protocol, aimed at solving online the turns determination problem among the agents. In this view, the turn determination problem can be seen as a distributed resource allocation problem. More in detail, the key idea is to consider the turn management problem as a drinking philosophers problem (DrPP) [23], which is a resource allocation problem for distributed and concurrent systems and is an extension of the well-known dining philosophers problem [24]. In this context, a DrPP-based algorithm can be used to solve the turn management problem. This type of solution can be implemented in a distributed fashion and has desirable characteristics like deadlock-freeness and fairness. The token based algorithm can be seen also as an approach for solving the minimal coloring problem for undirected graphs in a distributed way. Performed tests show that, although sub-optimal, it is however able to color a graph with a minimum number of colors in many scenarios.

II. Mathematical Preliminaries

Definition 2.1: (**Graph**): A graph Γ is an ordered pair $\Gamma(\mathcal{A}, \mathcal{E})$, such that

- $\mathcal{A} \subset \mathbb{Z}_+$ is the set of N nodes;
- $\mathcal{E}$ is a subset of pairs of $\mathcal{A}$, aka the set of edges connecting two nodes, i.e. $\mathcal{E} \subset \mathcal{A} \times \mathcal{A}$.

Definition 2.2: **(Adjacency Matrix)**: The adjacency matrix A of $\Gamma(\mathcal{A}, \mathcal{E})$, indexed by $\mathcal{A}$, is the $n \times n$ matrix whose (i, j)-entry is defined as

$$A_{i,j} = \begin{cases} 1 & \textit{if } (i,j) \in \mathcal{E}, \\ 0 & \textit{if } (i,j) \notin \mathcal{E}. \end{cases}$$

Definition 2.3: **(Degree Matrix)**: The degree matrix D of $\Gamma(\mathcal{A}, \mathcal{E})$ is the $n \times n$ matrix defined as

$$D_{i,j} = \begin{cases} \textit{deg}(i) & \textit{if } (i=j), \\ 0 & \textit{otherwise}. \end{cases}$$

where the degree $\textit{deg}(i)$ of a vertex counts the number of times an edge terminates at that vertex.

Definition 2.4: (**Pontryagin-Minkowski Difference**): Given two sets $\mathcal{B}_1, \mathcal{B}_2 \in \mathbb{R}^n$, their Pontryagin-Minkowski Difference is defined as

$$\mathcal{B}_1 \sim \mathcal{B}_2 := \{b_1 | b_1 + b_2 \in \mathcal{B}_1, \forall\, b_2 \in \mathcal{B}_2\}.$$

Definition 2.5: (**Neighborhood of the i-th node**): The neighborhood $\mathcal{N}_i$ of the i-th node in $\Gamma(\mathcal{A}, \mathcal{E})$, with cardinality $n_i = card(\mathcal{N}_i)$, consists of all its adjacent nodes i.e.

$$\mathcal{N}_i = \{i\} \cup \{j \in \mathcal{A} : (i,j) \in \mathcal{E}\}. \tag{1}$$

Definition 2.6: (**Turn**) A turn $\mathcal{T} \subset \mathcal{A}$ is a *subset of non-neighboring nodes*, i.e. $\forall i, j \in \mathcal{T}$ such that $i \neq j$, $j \notin \mathcal{N}_i$ (none of them is a neighbor of the others).

Definition 2.7: (**Concatenation Operator**) Given two sequences $\tau_1 := \{\tau_1^1, ..., \tau_1^n\} \in \{0,1\}^n$ and $\tau_2 := \{\tau_2^1, ..., \tau_2^m\} \in \{0,1\}^m$, the concatenation operator ("$\boxplus$") joins the two sequence values end to end, i.e.

$$\tau_1 \boxplus \tau_2 := \{\tau_1^1, \ldots, \tau_1^n, \tau_2^1, \ldots, \tau_2^m\} \in \{0,1\}^{n+m}. \tag{2}$$

III. Background

Consider a set of N subsystems $\mathcal{A} = \{1, \ldots, N\}$. Each subsystem is assumed to be a LTI discrete-time closed-loop dynamical system regulated by a local controller that ensures asymptotic stability. Let the i-th subsystem be described by the following model

$$\Sigma_i : \begin{cases} x_i(t+1) &= \Phi x_i(t) + G g_i(t), \\ c_i(t) &= H_i^c x_i(t) + L_i g_i(t), \end{cases} \tag{3}$$

where: $t \in \mathbb{Z}_+$, $x_i \in \mathbb{R}^n$ is the state vector (which includes the controller states under dynamic regulation), $g_i \in \mathbb{R}^2$ the manipulable command vector useful to track the local nominal reference sequence $r_i \in \mathbb{R}^2$, since planar tracking scenarios are considered, and matrices $\Phi \in \mathbb{R}^{n \times n}$ and $G \in \mathbb{R}^{n \times m}$ are the same for all agents. The state can be decomposed as $x_i(t) = [p_i(t), \sigma_i(t)]^T$, where $p_i(t) = [p_i^x(t), p_i^y(t)]^T$ represents the position of agent i in the earth-fixed frame and $\sigma_i(t)$ contains the remaining components of the state (such as velocity and controller states). Moreover, $c_i \in \mathbb{R}^{z_i}$ collects all those quantities that should be constrained within a local scope. More in details, each subsystem has to comply with *Local* and *Global* constraints. Hereafter, the terms subsystem, agent and vehicle will refer to the same entity. Furthermore, as already mentioned, it is assumed that each subsystem represents a closed-loop dynamic regulated by a decentralized local controller that guarantees i) asymptotic stability and ii) offset-free property (i.e. $(I_n - \Phi)^{-1} G = I_n$).

A. Convex Constraints

Vehicles have to satisfy *convex constraints*, which can be related to local constraints, i.e. speed limit, controller input saturation, and coupling constraints, i.e. proximity keeping. To this end, let the aggregate constrained vector $c = [c_1^T, \ldots, c_N^T]^T \in \mathbb{R}^z$, with $z = \sum_{i=1}^N z_i$ be defined as well. It is required that at each time instant $c(t)$ must be constrained in the set $\mathcal{C}$ being a specified convex and compact set described by n_q inequalities based on the continuous functions $\gamma_k : \mathbb{R}^z \to \mathbb{R}$, i.e.

$$\mathcal{C} := \{c \in \mathbb{R}^z : \gamma_k(c) \leq 0,\, k = 1, \ldots, n_q\}. \tag{4}$$

Then, the distributed CG design problem can be stated as follows

Problem *1: Given systems* (3) *and constraints* (4)*, locally determine, at each time step t and for each agent $i \in \mathcal{A}$, a suitable command signal $g_i(t)$ that is the best approximation of $r_i(t)$ and such that its application does not lead to constraint violations, i.e. $c(t) \in \mathcal{C}$, $\forall t \in \mathbb{Z}_+$.*

In order to recast constraints of Problem 1 in a form that is more akin to be managed within a distributed CG-based supervision strategy, some definitions are introduced.

First, let the steady-state constrained local output be defined as

$$c_{g_i} := H_i^c (I - \Phi_i)^{-1} G_i g_i,$$

and the *virtual* evolutions be

$$c_i(k, x_i, g_i) := H_i^c \left(\Phi_i^k x_i + \sum_{\tau=0}^{k-1} \Phi_i^{k-\tau-1} G_i g_i \right) + L_i g_i.$$

Notice that c_i is computed along a *virtual* time k starting from the initial state x_i corresponding to the application of a constant command sequence $g_i(t) \equiv g_i$ to Σ_i. Each c_{g_i} and $c_i(k, x_i, g_i)$ only depends on the local state and command g_i.

Secondly, it is important to note that, in certain instances (e.g., for systems with minimal or weak dynamic interaction and coupling constraints), each constraint function γ_k within $\mathcal{C}$ pertains to a restricted subset of agents only. In view of this, it is worth emphasizing that the above constraints structure can be associated to an undirected connected graph $\Gamma(\mathcal{A}, \mathcal{E})$, whose set of nodes coincides with the set of agents $\mathcal{A}$ and where $\mathcal{E} \subset \mathcal{A} \times \mathcal{A}$ is the set of edges connecting agents (nodes) whose subsystem evolutions are jointly constrained, i.e.

$$\begin{aligned} \mathcal{E} := \{(i,j) : \gamma_k \text{ in (4) involves both } i \text{ and } j \\ \text{for at least } k \in \{1, \ldots, n_q\}\}. \end{aligned} \tag{5}$$

It is assumed that neighboring agents are capable to share information with each other. Then, from the perspective of the i-th agent, the following quantity, which collects all local constrained variables, becomes relevant

$$c_{[i]} := [c_{i_1}^T, c_{i_2}^T, \ldots, c_{i_h}^T, \ldots, c_{i_{N_i}}^T] \in \mathbb{R}^{z_{[i]}} \tag{6}$$

with $z_{[i]} := \sum_{i \in \mathcal{N}_i} z_i$. Please notice that the quantities $x_{[i]} \in \mathbb{R}^{n_{[i]}}$ and $g_{[i]} \in \mathbb{R}^{m_{[i]}}$ can be analogously defined, while the quantities $\tilde{x}_i \in \mathbb{R}^{n_{[i]}-1}$ and $\tilde{g}_i \in \mathbb{R}^{n_{[i]}-1}$, collect, respectively, all states and commands of the neighbors of the $i-th$ agent. In a similar way, let us recast each function γ_k in (4) involving agent i and its neighbors only as $\gamma_{[i]}^K : \mathbb{R}^{z_{[i]}} \to \mathbb{R}$. Then, the constraint set (4) can be seen as a combination of several compact and convex sets defined as

$$\mathcal{C}_i := \{c_{[i]} \in \mathbb{R}^{z_{[i]}} : \gamma_{[i]}^h(c_{[i]}) \leq 0,\, h = 1, \ldots, n_{q_i}\}. \tag{7}$$

B. Turn-Based Distributed Command Governor

Let us consider a series of L *turns* $\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_L$ such that $\bigcup_{l=1}^L \mathcal{T}_l = \mathcal{A}$ and let us assume that every agent knows its belonging turn. According to the TB-CG algorithm, at time t, only the agents belonging to the current turn, say it $\mathcal{T}_{l(t)}$, can update their local commands $g_i(t)$ whereas all other agents in

$\mathcal{A} \setminus \mathcal{T}_{l(t)}$ have to keep applying the local command they have applied at time $t-1$. For this reason, after each agent in $\mathcal{T}_{l(t)}$ received from its neighbors the values of their states and of their previously applied commands, i.e. $x_{[i]}(t)$ and $g_{[i]}(t-1)$, the global constraints could be satisfied if the local command $g_i(t)$ were selected as

$$g_{[i]}(t) \in \mathcal{V}_i(x_{[i]}(t)), \tag{8}$$

where

$$\mathcal{V}_i(x_{[i]}) := \left\{ g_{[i]} \in \mathcal{W}_i | c_{[i]}(k, x_{[i]}, g_{[i]}) \in \mathcal{C}_i,\ \forall k \in \mathbb{Z}_+ \right\}, \tag{9}$$

with

$$\mathcal{W}_i := \{ g_{[i]} \in \mathbb{R}^{m_{[i]}} :\ c_{g_{[i]}} \in \mathcal{C}_i \sim \mathcal{B}_\delta \} \tag{10}$$

representing the set of steady-state admissible commands, while $g_{[i]}(t)$ is set equal to $g_{[i]}(t-1)$ for all entries except g_i. Notice that for a static configuration of the network the turns $\mathcal{T}_l$ are *L-periodic*, meaning that $\mathcal{T}_{l(t+L)} = \mathcal{T}_{l(t)}$. This concept can be formalized as follows

Algorithm 1: The Turn-Based CG (TB-CG)

(Agent i) repeat at each time t

1: **if** $i \in \mathcal{T}_{l(t \mod L)}$ **then**
2: receive $\tilde{x}_i(t), \tilde{g}_i(t-1)$ from neighbours;
3: solve

$$g_i(t) = \arg\min_{g_i} ||g_i - \hat{r}_i(t)||^2_{\Psi_i} \tag{11}$$

$$\text{subject to } (8)$$

4: **else**
5: set $g_i(t) = g_i(t-1)$ and apply $g_i(t)$
6: transmit $g_i(t)$ and $x_i(t)$ to the neighbouring agents
7: **end if**

where $\Psi_i = \Psi_i^T > 0, \forall i \in \mathcal{A}$ and $\hat{r}_i(t) \equiv r_i(t)$ unless stated otherwise.

Remark 1: From a theoretical perspective, a system in the form of (3) can represent various classes of systems, including non-holonomic and underactuated vehicles [28]. Consequently, the CG strategy can be directly applied to such class of vehicles, provided that a linear time-invariant model accurately captures these vehicle characteristics is available.

C. Plug & Play (PnP) Prescriptions

In [21], a generalization of **Algorithm 1** was introduced in order to deal with online constraints modification by means of *plug-in* and *plug-out* operations. In this way, it is possible to include and remove dynamically *nodes* from an existing network.

In this context, it is necessary to deal with the following *Plug & Play* (*PnP*) scenarios:

Scenario *1: Edge addition* (*plug-in*): At a certain time instant $t^{s_1}_{start} > 0$, an agent j associated to the subsystem Σ_j, with a set of neighbors $\mathcal{N}_j$, requests the permission to agent $i \in \mathcal{A}$ to share some constraints at time $t^{s_1}_{end} \geq t^{s_1}_{start}$. This requires that at time $t^{s_1}_{end}$ the following joining process must be completed

i. $\mathcal{E} \leftarrow \mathcal{E} \cup \{(i,j)\} \cup \{(j,k)\, \forall k \in \mathcal{N}_j\}$;
ii. $\mathcal{N}_i \leftarrow \mathcal{N}_i \cup \{j\}$;
iii. $\mathcal{N}_j \leftarrow \mathcal{N}_j \cup \{i\}$;
iv. $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{N}_j$.

Scenario *2: Edge removal* (*plug-out*): At a certain time instant $t^{s_2}_{start} > 0$, an agent $j \in \mathcal{A}$ requests to agent i the permission to remove some shared constraints at time $t^{s_2}_{end} \geq t^{s_2}_{start}$. This requires that the following disjointing process should be completed at time $t^{s_2}_{end}$

i. $\mathcal{E} \leftarrow \mathcal{E} \setminus \{(i,j)\}$;
ii. $\mathcal{N}_j \leftarrow \mathcal{N}_j \setminus \{i\}$;
iii. $\mathcal{N}_i \leftarrow \mathcal{N}_i \setminus \{j\}$.

Note that in Scenario 2, if $\mathcal{N}_i = \emptyset$ then i needs to be removed from $\mathcal{A}$ as well.

Under the assumption that each agent is supervised by a local CG module, the above operations are performed by modifying the local CG$_i$ units. When a node j, which can be internal $j \in \mathcal{A}$ or external $j \notin \mathcal{A}$, requests a *plug-in* operation to an agent $i \in \mathcal{A}$, the new *coupling* constraints set has to be included

$$\begin{aligned} \mathcal{C}_{i,j} := \{ &[c_i^T, c_j^T]^T \in \mathbb{R}^{z_i + z_j} : \gamma^h_{i,j}(c_{[i]}) \leq 0, \\ &h = 1, ..., n_{q_{i,j}} \}, \end{aligned} \tag{12}$$

where $\gamma^h_{i,j}(c_{[i]})$ depends also on the coupling constraints between agent i and j. When the new node j joins the existing network, $\mathcal{C}_i$ and $\mathcal{C}_j$ are replaced by the following sets respectively

$$\begin{aligned} &\mathcal{C}_i^{new} := \{ [c^T_{[i]}, c_j^T]^T \in \mathbb{R}^{z_{[i]} + z_j} : c_{[i]} \in \mathcal{C}_i, \\ &[c_i^T, c_j^T]^T \in \mathcal{C}_{i,j} \}, \\ &\mathcal{C}_j^{new} := \{ [c_j^T, c_i^T]^T \in \mathbb{R}^{z_i + z_j} : c_j \in \mathcal{C}_j, \\ &[c_i^T, c_j^T]^T \in \mathcal{C}_{i,j} \}. \end{aligned} \tag{13}$$

As a consequence

$$\begin{aligned} \mathcal{C}^{new} := \{ &[c^T, c_j^T]^T \in \mathbb{R}^{z + z_j} : c \in \mathcal{C}, \\ &[c_i^T, c_j^T]^T \in \mathcal{C}_{i,j} \}. \end{aligned} \tag{14}$$

In the case of *plug-out* operations, when the node j is removed, all sets $\mathcal{C}_i$ for which $i \in \mathcal{N}_j$ are replaced with sets $\mathcal{C}_i^{new}$ constructed from $\mathcal{C}_i$ by deleting inequalities involving $\gamma^h_{[j]}, h = 1, ..., n_{q_j}$, and the same reasoning holds for $\mathcal{C}^{new}$.

In view of the above considerations, Problem 1 needs to be solved concurrently with this further problem:

Problem *2:* Given systems (3) and constraints (4), whenever either Scenario 1 or 2 occurs, compute the local command $g_i(t)$ to guarantee a safe transition from $\mathcal{C}$ to $\mathcal{C}^{new}$, i.e. $c(t) \in \mathcal{C}, \forall t \in [t^\bullet_{start}, t^\bullet_{end} - 1]$ and $[c^T(t), c_j^T(t)]^T \in \mathcal{C} \leftarrow \mathcal{C}^{new}, \forall t \geq t^\bullet_{end}$ (the bullet is related to a possible PnP scenario 1 or 2).

Definitions and conditions for *plug-in* operations have been defined in [22]. Thus, the most relevant concepts are here reported for clarity.

Definition 3.1: Two nodes i and j are *pluggable* at time t, i.e. *plug-in* operation envisaged in Scenario 1 can be safely performed, if the current state pair $c_i(k, x_i(t), g_i(t-1)), c_j(k, x_j, g_j(t-1)) \in \mathcal{C}_{i,j}$.

Since each system is supervised by a local CG, it is reasonable to assume that a *plug-in* operation always occurs in the case where current state and reference can be steered to feasible equilibria w.r.t. the constraints sets. If the nodes are *pluggable* at time t, then such an operation is performed at

time t. If not, agents i and j need to be conducted towards a *pluggable* condition.

Definition 3.2: At time t_0 a node j can ask its neighbors to stop updating their commands $\tilde{g}_{[j]}$ indefinitely, i.e.

$$\hat{r}_\eta(t) \equiv g_\eta(t_0 - 1) \quad \forall \eta \in \mathcal{N}_j \setminus \{j\},\ t \geq t_0 \tag{15}$$

executing a *freeze* operation. Conversely, j can ask its neighbors to resume their default condition by performing an *unfreeze* operation.

In order to ensure that a *plug-in* operation can be completed at time t, in [21] it was shown that the entire network needs to transit toward constraints compliant equilibria computed as a solution of the following problem

$$\begin{bmatrix} r_i^\star \\ r_j^\star \end{bmatrix} = \underset{w_i, w_j}{\arg\min} \|w_i - g_i(t-1)\|_{\Psi_i} + \\ + \|w_j - g_j(t-1)\|_{\Psi_j} \tag{16}$$

$$s.t. \quad [w_i^T, \tilde{g}_{[i]}^T(t-1)]^T \in \mathcal{W}_i^\delta, \tag{17}$$

$$[w_j^T, \tilde{g}_{[j]}^T(t-1)]^T \in \mathcal{W}_j^\delta, \tag{18}$$

$$[w_i^T, w_j^T]^T \in \mathcal{W}_j^{\delta,new}. \tag{19}$$

If the previous problem has a solution, agents i and j ask to their respective neighbors to *freeze* their commands $\tilde{g}_{[i]}(t)$ and $\tilde{g}_{[j]}(t)$ until the i-th and j-th agents become pluggable. The references $\hat{r}_i \equiv r_i^\star$ and $\hat{r}_j \equiv r_j^\star$ are considered in place of $r_i(t)$ and $r_j(t)$ respectively in the local CG optimization problem (11).

As a consequence, Plug & Play requests are processed before Algorithm 1 is executed, and virtual references in (16) are generated for agents that are not *pluggable*. Possible turns reconfigurations are entrusted to a Turn Manager Unit, that will be described later, which receives notifications after either virtual reference or admissible command computations and decides when agents can update their admissible command. In this respect, **Algorithm 1** needs to be extended in the form of **Algorithm 1a**. Please note that, in order to manage and process multiple requests efficiently, a data structure *ppStack* needs to be introduced.

Algorithm 1a: The Turn-Based CG (TB-CG) with Plug & Play Capabilities

(Agent i) Initialize
$ppStack = []$;
execute for each time t
if $i \in \mathcal{T}_{l(t)}$ **then**
  if PnP request is needed with j **then**
    send $ppReq_i$ to j;
    freeze neighbors;
    wait for instructions from j and execute them;
    unfreeze neighbors;
  end if
  if $ppStack$ is not empty **then**
    pop $ppReq_j$ from $ppStack$;
    if $ppReq_j$ is a *plug-in* request **then**
      check if i and j are *pluggable* at time t;
      if not *pluggable* **then**
        freeze neighbors;
        compute virtual references $r_i^\star$, $r_j^\star$ by solving Problem (16-19);
        ask j to consider $\hat{r}_j \equiv r_j^\star$ until *pluggable*;
        consider $\hat{r}_i \equiv r_i^\star$ and apply until *pluggable*;
        notify end of command computation;
        unfreeze neighbors;
      end if
      add new shared constraints;
      set $\mathcal{N}_i \leftarrow \mathcal{N}_i \cup \{j\}$;
    else
      remove old shared constraints;
      set $\mathcal{N}_i \leftarrow \mathcal{N}_i \setminus \{j\}$;
    end if
    notify PnP acceptance to j;
  end if
  execute Algorithm 1;
  notify end of command computation;
end if

IV. Problem Formulation

Because vehicles may operate in restricted areas and their trajectories may intersect during their actions, adding *Collision Avoidance* constraints becomes crucial. To deal with this type of constraints, Problem 1 and Problem 2 are extended including the following constraints

$$||p_i(t) - p_j(t)||_\infty \geq d_{min},\ \ \forall t \geq 0 \tag{20}$$

where d_{min} represents the minimum distance vehicles have to maintain among them in order to avoid collisions. Please notice that this type of constraint introduces *coupling*, because the evolution between subsystems are jointly constrained. More in detail, by analyzing problem (16), it is noticeable that only steady-state constraints could be considered during the computation of the virtual references $[r_i^\star, r_j^\star]$. For this reason, it stands clear that a feasible solution could guarantee *Collision Avoidance* at steady-state but not during transients.

Problem *3: Solve Problems* 1 *and 2 including Collision Avoidance constraints in the form (20).*

V. Supervision Architecture

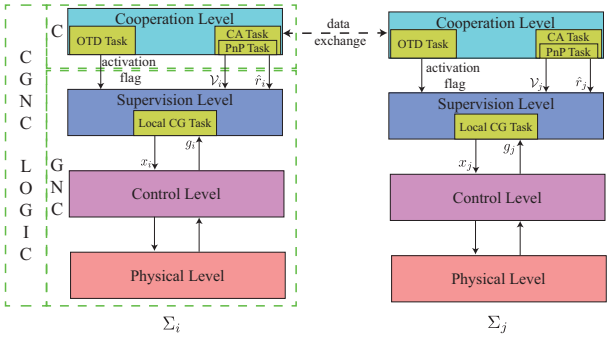


Fig. 1. Enhanced multilayer supervision architecture.

In this section, we basically face two challenges: i) ensuring the feasibility of previous feasible solutions for PnP problems when non-convex collision avoidance constraints are added, and ii) guaranteeing that a turn re-determination procedure is performed. More specifically, when a *plug-in* or a *plug-out* request is carried out, a turn re-determination procedure

should take place. In order to solve Problem 3, a further extension, to the distributed supervision architecture proposed in [21], is introduced. More in detail, each vehicle employs the multilayered architecture illustrated in Figure 1, where the first three layers encapsulate functionalities associated with Guidance, Navigation, and Control (GNC). Specifically, the *Physical Level* and the *Control Level* represent the pre-compensated system Σ_i, while the *Supervision Level* integrates the CG strategy (*Local CG* task). Additionally, the uppermost layer, i.e. the *Cooperation Level*, incorporates cooperative (C) functionalities, including turn management (the *OTD* task, which enables the execution of the CG task through an activation flag) and dynamic constraints management (the *PnP* and *CA* tasks). It is important to note that, encapsulating the *PnP* task within the *CA* task ensures full compatibility with the architecture proposed in [21], while also enabling dynamic collision avoidance constraint management.

A. PnP Task: Collision Avoidance Inclusion

To ensure the feasibility of the dynamic inclusion of *Collision Avoidance* constraints, a collection of circular regions has been defined (see Figure 2).

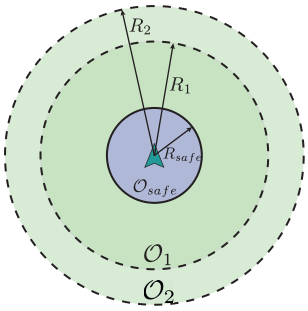


Fig. 2. Operative regions defined around a vehicle.

Each region is centered at the vehicle location and is included to address a specific purpose. The innermost region, named *Safe Area* and defined by the radius $R_{safe} > d_{min}$, identifies a circle $\mathcal{O}^i_{safe}(x_i(t))$ that can be formalized as follows

$$\mathcal{O}^i_{safe}(x_i) := \{g_i \in \mathbb{R}^2 \mid \|p_i - g_i\| \leq R_{safe}\}, \tag{21}$$

where R_{safe} denotes the maximum admissible distance between the current position of the vehicle p_i, contained in the state vector x_i, and the command g_i to be computed.

The dimension of the region $\mathcal{O}_{safe}$ is constant and assigned from the outset. To provide further insight into the tuning of R_{safe}, selecting higher values results in a broader admissible region where the CG module can search for feasible commands. However, larger values of R_{safe} require a more conservative approach to the plug-in (PnP) operation, which must be initiated further in advance. An intuitive strategy for selecting R_{safe} is to choose the smallest possible value satisfying $R_{safe} > d_{min}$, while maximizing a given performance index, such as vehicle speed. In the following, we provide the conditions that guarantee agents which have not yet performed a PnP operation—and therefore do not share collision avoidance constraints—will not compute their admissible commands within each other's *Safe Areas*, hence the term *Safe*. The intermediate region $\mathcal{O}^i_1(p_i, R_1) := \{g_i \in \mathbb{R}^2 \mid \|p_i - g_i\| \leq R_1\}$ denotes the safety region out of which a *plug-in* request can take place without violating the *Collision Avoidance* constraints. In practice, a plug-in request at time t from an agent j can be conveyed to agent i while

$$p_j(t) \notin \mathcal{O}^i_1(p_i(t), R_1). \tag{22}$$

The region defined as $\mathcal{O}^i_2(p_i, R_2) := \{g_i \in \mathbb{R}^2 \mid \|p_i - g_i\| \leq R_2\}$ denotes the largest region where it is mandatory to share a *Collision Avoidance* constraint with another vehicle. A *plug-out* operation is performed whenever the relative distance between two vehicles, sharing *Collision Avoidance* constraints, exceeds the R_2 threshold, i.e. $p_j(t) \notin \mathcal{O}^i_2(p_i, R_2)$.

An important role is played by the parameter R_1. In fact, it will be proved in Section 5 that a collision is always avoided during a plug-in operation, if it is larger than a minimum value $R^\star_1$ defined as

$$R^\star_1 := 2R_{Safe} + d_{min} + \varepsilon, \tag{23}$$

where $\varepsilon > 0$ is a positive scalar that takes into account possible overshooting effects of the pre-compensated system dynamics. Figure 3 illustrates the graphic idea underlying equation (23)

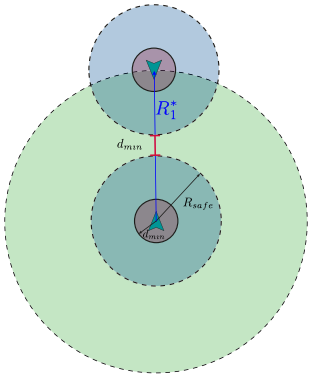


Fig. 3. Graphical representation of the distance when a *plug-in* request is triggered.

and the distance $R^\star_1$. Let us consider a scenario where a *plug-in* operation requires the resolution of problem (16-19). Please notice that condition $R_1 > R^\star_1$, combined with the fact that $r^\star_i \in \mathcal{O}^i_{safe}(x_i)$ and $r^\star_j \in \mathcal{O}^j_{safe}(x_j)$, where $r^\star_i$ and $r^\star_j$ are computed by solving (16-19), ensures that while applying $r^\star_i$ and $r^\star_j$, *Collision Avoidance* constraints are never violated.

Please notice that, thanks to these operative regions, collisions between neighbors of agent i and agent j (that have their respective commands frozen) cannot occur because agent i and agent j are the nearest agents of the two formations.

B. Turn Management Protocol

According to [21], a turn management algorithm should satisfy the following properties:

- two neighboring agents should not belong to the same turn;

- all agents of the network are periodically selected, i.e. $\cup_{l=1}^{L}\mathcal{T}_l = \mathcal{A}$;
- the number of turns L should be as small as possible.

As reported in [30], turn management can be addressed solving well-known minimal vertex coloring problem, i.e., to determine an assignment color (turn) to each graph vertex (vehicle) such that no two adjacent vertices have the same color, and the number of colors is minimal. Due to the fact that the minimal vertex coloring problem is an NP-complete optimization problem [25] and due to the complexity of the distributed version of the algorithm [26], an innovative heuristic turn management strategy which satisfies the three previous properties is proposed. The presented algorithm is a token-based protocol, based on a solution of the well-known drinking philosophers problem (DrPP) [23], which intrinsically enforces a turn mechanism based on the following simple principles:

1) If an agent i has no neighbors, it can update the command g_i at each time instant without violating constraints.
2) When only agents i and j are connected, a shared token (associated to that connection) between them is created and codified as a function $\tau_i^j : \mathbb{R} \to \{0,1\}$ defined as

$$\tau_i^j(t) := \begin{cases} 1, & \text{if the token from j is held.} \\ 0, & \text{otherwise.} \end{cases} \quad (24)$$

In this situation, agent i can update its admissible command only if it holds the token. After the computation, in order to allow a change of turn, the agent i drops the token to its neighbor j and so on. A direct consequence of definition (24) is that $\tau_j^i(t) = 1 - \tau_i^j(t)$.
3) When more neighbors are present, agent i must wait to hold all tokens from its neighbors before proceeding to the command updating. Then, if we denote with $\tau_i(t) := \{\tau_i^{i_1}, ..., \tau_i^{i_{n_i}}\} \in \{0,1\}^{n_i}$ the sequence of tokens held by agent i at time t, the condition allowing agent i to update its command g_i in (11) is

$$\tau_i(t) = \mathbf{1}_{n_i} \implies \tau_i^{i_i}(t) = 1, \quad i_i = i_1, \ldots, i_{n_i}. \quad (25)$$

4) Upon a new connection, agent j requesting a *plug-in* to agent i, it provides a new token $\tau_i^j(t+1) = 1$. The new token sequences become $\bar{\tau}_i(t+1) = \tau_i(t+1) \boxplus \tau_i^j(t+1) = \tau_i(t+1) \boxplus 1$ and $\bar{\tau}_j(t+1) = \tau_j(t+1) \boxplus \tau_j^i(t+1) = \tau_j(t+1) \boxplus 0$. Notice that $\bar{\tau}_i \in \{0,1\}^{n_i+1}$ and $\bar{\tau}_j \in \{0,1\}^{n_j+1}$.

A first main consequence of the above described rules involves the definition (2.6) of turn within the following Theorem

Theorem 1: At each time instant t, the set $\mathcal{T}(t) \subset \mathcal{A}$ of agents satisfying condition (25) is a turn.

Proof: It is enough to notice that, thanks to condition (24) for each $j \in \mathcal{N}_i$ and $i \in \mathcal{T}(t)$, $\tau_i^j(t) = 0$. Then, no neighbors of a generic agent $i \in \mathcal{T}(t)$ belong to $\mathcal{T}(t)$. As a consequence, $\mathcal{T}(t)$ satisfies the definition in (2.6). □

The above described rules have been translated into the following **Algorithm 2** that acts as Turn Manager Unit in **Algorithm 1a**.

Algorithm 2: Token Manager

(Agent i)
1: receive tokens from neighboring nodes;
2: **if** $\tau_i == 1_{n_i}$ **then**
3: $\quad \mathcal{T}_{l(t)} \leftarrow \mathcal{T}_{l(t)} \cup \{i\}$;
4: $\quad$ wait for admissible command computation;
5: $\quad \mathcal{T}_{l(t)} \leftarrow \mathcal{T}_{l(t)} \setminus \{i\}$;
6: $\quad$ transmit tokens to neighbors;
7: $\quad \tau_i \leftarrow 0_{n_i}$
8: **end if**

Notice that instruction 4 is a blocking instruction. The code execution resumes only after that a notification of command computation has been received (instructions 17 and 29 in **Algorithm 1a**). Please notice that the above algorithm is performed by each vehicle at each sampling time.

C. Multiple Plug&Play

The *plug-in/plug-out* strategy outlined so far does not restrict several *Plug & Play* operations from being requested. *Plug-out* procedures, either multiple and parallel, can be conducted safely since removing constraints has no effect on the feasibility of the optimization problems [21]. As a result, the emphasis here is made only on the adaptations needed to perform multiple and parallel *plug-in* operations safely. In

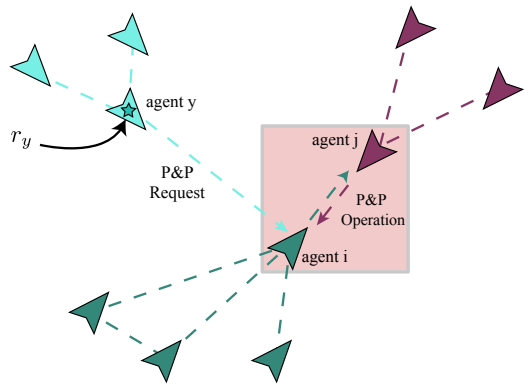


Fig. 4. Multiple *plug-in* scenario: agent y sends a request to agent i that is already in a *PnP* operation with agent j.

order to maintain the feasibility of the overall strategy, it is critical to handle one *plug-in* at a time. Each agent has a LIFO stack where it stores PnP requests. Let us consider a scenario where an agent requests to share constraints with another vehicle that is already engaged in a *plug-in* operation. In particular, an agent y sends a PnP request to an agent already involved into a PnP procedure (agent i in Figure 4). Because i and j are engaged in PnP maneuvers, the request coming from y cannot be handled immediately and it is marked as *pending*. As a consequence, agent y waits for its request to be elaborated, *freezing* its neighbors and itself using its current position as reference, i.e. it sets $\hat{r}_y \equiv p_y(t)$ and $\hat{r}_\eta \equiv g_\eta(t-1) \forall \eta \in \mathcal{N}_y$. Agent i, that is involved in PnP operation, pushes the request from agent y into its LIFO stack. The same happens if other requests are sent to agent i. After that agent i completes its PnP operation with agent j,

it retrieves and handles the PnP requests from its LIFO stack. This process continues until the stack is not empty.

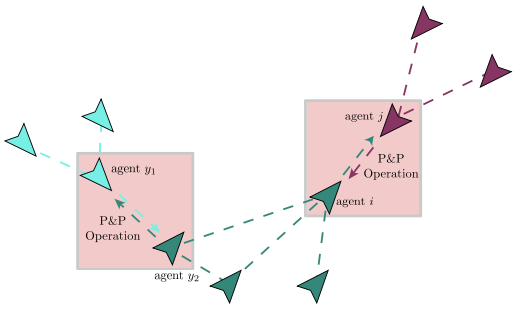


Fig. 5. Parallel *plug-in* scenario: agent y_2 and agent i, belonging to the same formation, perform a *PnP* operation with agent y_1 and agent j, respectively, that belong to different formations.

Please keep in mind that, due to the addition of the Operative regions, *Collision Avoidance* constraints are still feasible throughout the whole operation and it is not necessary to include them in Problem (16). Parallel *plug-in* operations may involve different vehicles of the same formation, as shown in Figure 5. Parallel *plug-in* operations associated to agents that are not directly involved in *plug-in* procedures (i.e. have their commands frozen) can be safely performed. In contrast, when an agent has its command frozen (agent y_2 in the figure) and receives a *plug-in* request, it handles it according to a LIFO policy by following the steps described above. The previously described reasoning may be summarized in the following algorithm.

Algorithm 3: Multiple and Parallel *PnP*

(Agent i) Initialize $ppStack_i \leftarrow [\,]$, $j_{curr} = 0$ and set busy to false;
if *plug-in* request received from j **then**
  if not busy **then**
    set busy to true;
    set $j_{curr} = j$;
  else
    push j onto $ppStack_i$;
    ask j to set $\hat{r}_j \equiv p_j(t)$, *freeze* its neighbors and set their busy state to true;
  end if
end if
if $j_{curr} \,!= 0$ **then**
  if agent i and j_{curr} *pluggable* **then**
    perform *plug-in* operation: *unfreeze* all neighbors and set their busy state to false;
    if $ppStack_i = [\,]$ **then**
      set busy to false and $j_{curr} = 0$;
    else
      pop the top of $ppStack_i$ and set j_{curr};
    end if
  else
    compute virtual references $[r_i^\star, r_{j_{curr}}^\star]$ solving (16) subject to (17-19);
    freeze i-th and j_{curr}-th neighbors;
  end if
end if

VI. Computational Details and Implementability

It is worth mentioning that set $\mathcal{C}$ apparently includes non-convex constraints (20). Because solving convex optimization problems is more convenient, a convexification procedure is introduced to reformulate the *Collision Avoidance* constraints (20), for the whole formation, into linear constraints depending on continuous and integer variables [27, 20]

$$\begin{array}{lllll} \forall i,j|i>j: & p_i^x - p_j^x & \geq & d_{min} - \mu b_{ij}^1 \\ \text{and} & p_i^y - p_j^y & \geq & d_{min} - \mu b_{ij}^2 \\ \text{and} & p_j^x - p_i^x & \geq & d_{min} - \mu b_{ij}^3 \\ \text{and} & p_j^y - p_i^y & \geq & d_{min} - \mu b_{ij}^4 \\ \text{and} & \sum_{k=1}^4 b_{ij}^k & \leq & 4-1, \end{array} \tag{26}$$

where $b_{ij}^k \in \{0,1\}$ represents a binary variable and μ a sufficiently large scalar [29].

Thanks to this reformulation, it is possible to easily include the *Collision Avoidance* constraints in the optimization problem, which becomes a mixed integer quadratic optimization problem (MIQP). To this end, numerous commercial and non-commercial solvers can be used to solve the optimization problem. In general, the time required to solve the above introduced optimization problems depends on several factors, such as the chosen numerical solver, the number of constraints, the hardware platform and the code optimization. In this work, the numerical Gurobi solver [33], which computes the optimal solution when feasibility is ensured, was employed in MATLAB. Further details regarding computational performance are provided below.

A. Computational Demands

In a preliminary study presented in [34], an evaluation of the computational and communication demands was conducted, comparing two distinct approaches: i) the approach presented in [19], which handled only static coupling constraints enforced among all agents from the outset, and ii) the proposed approach, which uses handles dynamic addition/removal of coupling constraints by *PnP* procedures. More in detail, using the same simulation environment, multiple simulations with an increasing number of agents (up to 18) within a confined space, designed to increase potential collisions, were conducted to evaluate the benefits of the scheme, in terms of scalability improvements, derived from dynamic constraint management. The quantitative improvements in CPU time for solving the optimization problem using dynamic constraints, as compared to static constraints, are detailed in Table I. Furthermore, Figure 6 illustrates the reduction in communication overhead, depicting the number of packets exchanged using predefined static constraints (blue) and dynamic constraints (green) when considering 18 agents. Simulation recordings demonstrating the behavior of agents using both predefined static and dynamically handled constraints are available at https://youtu.be/Kscv6e3JjGo and https://youtu.be/MHCPZVjr914, respectively.

B. Implementability

The practical implementability of the proposed dynamic approach is supported by the successful implementation of the static approach [19] on real-world systems [35]. More in detail, the supervision scheme was employed to coordinate marine surface vehicles where predefined coupling constraints were enforced. The results demonstrated the suitability of the

N. of Agents		min	avg	worst
$N = 12$	**Static**	$0.0022\,[s]$	$0.0104\,[s]$	$0.0224\,[s]$
	Dynamic	$0.0014\,[s]$	$0.0087\,[s]$	$0.0132\,[s]$
	Improvement	36.36%	16.35%	41.07%
$N = 15$	**Static**	$0.0023\,[s]$	$0.0107\,[s]$	$0.0243\,[s]$
	Dynamic	$0.0015\,[s]$	$0.0090\,[s]$	$0.0143\,[s]$
	Improvement	34.78%	15.89%	41.15%
$N = 18$	**Static**	$0.0025\,[s]$	$0.0114\,[s]$	$0.0275\,[s]$
	Dynamic	$0.0015\,[s]$	$0.0091\,[s]$	$0.0172\,[s]$
	Improvement	39.36%	20.18%	37.45%

TABLE I
MEASURED CPU TIMES WITH STATIC AND DYNAMIC CONSTRAINTS AND PERCENTAGE IMPROVEMENT. RESULT WERE OBTAINED USING A WORKSTATION EQUIPPED WITH AN 8-CORE AMD RYZEN 7 5800X3D (@4.5 GHZ).

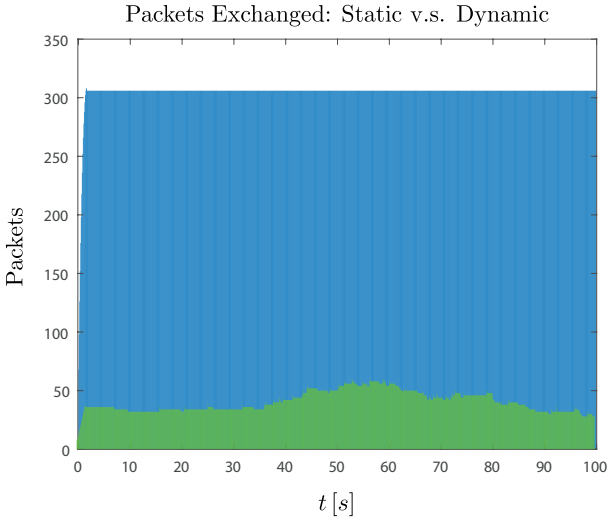


Fig. 6. Packets exchanged at each time instant using both predefined static (blue) and dynamic constraints (green) in an experiment involving 18 agents.

CG strategy and indicated that the computational complexity is manageable for typical vehicle systems. Given that the proposed approach enhances computational efficiency by dynamically adding and removing coupling constraints through *PnP* procedures, supported by the simulation results presented earlier, it is reasonable to expect that the proposed dynamic approach will meet the computational requirements imposed by real vehicles equipped with standard onboard hardware with improved, or at least comparable, performance.

In general, while factors such as modeling, identification, and validation accuracy are crucial determinants of overall performance, they are not inherent limitations of the dynamic approach itself. Another key consideration for successful real-world integration is the need to preserve the modularity of the strategy. For the practical deployment of the proposed approach within a Guidance, Navigation and Control (GNC) architecture, a vision module with sufficient range will be necessary to measure inter-vehicle distances. To facilitate a seamless transition from simulation to practical testing, the proposed distributed supervision scheme can be implemented into the Robot Operating System (ROS) environment [36], while the optimization problem can be modeled using the Python-embedded modeling language CVXPY [37] to allow the use of the same numerical solver Gurobi.

VII. PROPERTIES

In this section some properties of the proposed Supervision Architecture are presented. To this end, the following preliminary result is relevant

Lemma 7.1: If a *plug-in* request is sent at time $t = t_{start} > 0$ by an agent j to an agent $i \in \mathcal{A}$ with

$$p_j(t) \in \partial(\mathcal{O}_1(p_i(t), R_1^\star)), \ (R_1^\star \text{ defined as in (23)}) \tag{27}$$

then, *Collision Avoidance* constraints (20) are fulfilled by applying $[r_i^\star, r_j^\star]$, solution of problem (16).

Proof. Let's assume that at time $t_{start} > 0$ the current states $[x_i, x_j]$ need to be steered to a feasible equilibrium $[x_i^\star, x_j^\star]$ by means of applying the solution $[r_i^\star, r_j^\star]$ of problem (16) where *Collision Avoidance* constraints (20) are not included. When the *plug-in* request occurs, the relative position between the two agents is:

$$||p_i(t_{start}) - p_j(t_{start})|| \geq 2R_{Safe} + d_{min} + \varepsilon, \tag{28}$$

with $\varepsilon > 0$. In the worst case, $r_i^\star \in \partial(\mathcal{O}^i_{safe}(x_i(t_{start})))$ and $r_j^\star \in \partial(\mathcal{O}^j_{safe}(x_j(t_{start})))$, where $\partial(\mathcal{O}^i_{safe}(x_i))$ and $\partial(\mathcal{O}^j_{safe}(x_j)$ are the boundaries of $\mathcal{O}^i_{safe}(x_i)$ and $\mathcal{O}^j_{safe}(x_j)$ respectively. Then, it exists a finite time $t^\star > t_{start}$ such that

$$||p_i(t) - r_i^\star|| \leq \varepsilon/2 \quad \forall t \geq t^\star, \tag{29}$$

and

$$||p_j(t) - r_j^\star|| \leq \varepsilon/2 \quad \forall t \geq t^\star. \tag{30}$$

On the other hand, we have

$$||r_i^\star - r_j^\star|| \geq d_{min} + \varepsilon. \tag{31}$$

Then, the following inequalities hold true $\forall t \geq t^\star$

$$\begin{aligned}
||r_i^\star - p_i(t) + p_i(t) - r_j^\star|| &\geq d_{min} + \varepsilon \\
||r_i^\star - p_i(t)|| + ||p_i(t) - r_j^\star|| &\geq d_{min} + \varepsilon \\
\varepsilon/2 + ||p_i(t) - r_j^\star|| &\geq d_{min} + \varepsilon \\
||p_i(t) - r_j^\star|| &\geq d_{min} + \varepsilon/2 \\
||p_i(t) - p_j(t) + p_j(t) - r_{N+1}^\star|| &\geq d_{min} + \varepsilon/2 \\
||p_i(t) - p_j(t)|| + ||p_j(t) - r_j^\star|| &\geq d_{min} + \varepsilon/2 \\
||p_i(t) - p_j(t)|| + \varepsilon/2 &\geq d_{min} + \varepsilon/2
\end{aligned}$$

and the relative distance between agent i and agent j is going to be

$$||p_i(t) - p_j(t)|| \geq d_{min}, \quad \forall t \geq t^\star. \tag{32}$$

□

Theorem 2: Consider systems (3) along with the proposed distributed supervision architecture. Let Algorithm 2 be executed by agents in $\mathcal{A}$, distributed into turns $\mathcal{T}_l$ such that, periodically, all agents of the network are selected, i.e.

$$\exists t' : \forall t > 0, \cup_{i=0}^{t'} \mathcal{T}_{t+i} = \mathcal{A}.$$

Assume also that $c_i(k, x_i(0), g_i(0)) \in \mathcal{C}_i$ holds true. Then, any *plug-in* operation between two *pluggable* nodes i and

j, initiated at time $t^{s_1}_{start} > 0$, is performed in a finite time without any constraints violation, i.e.

$$\exists\ t^{s_1}_{end} < \infty : c(t) \in \mathcal{C},\ \forall t \in [t^{s_1}_{start}, t^{s_1}_{end} - 1],$$

and

$$[c^T(t), c^T_j(t)]^T \in \mathcal{C} \leftarrow \mathcal{C}^{new},\ \forall t \geq t^{s_1}_{end}\,.$$

Proof. The proof that the multi-layer supervision strategy solves Problem 3 without *Collision Avoidance* constraints is reported in [21]. Following the result of Lemma 7.1, the introduction of *Collision Avoidance* constraints (20) does not alter the properties of the solution. □

Finally further properties concerning feasibility, stability and convergence of the described supervision architecture are summarized in the following Theorem:

Theorem 3: Consider systems (3) and assume that **Algorithm 1a** is performed by agents in $\mathcal{A}$, distributed into turns $\mathcal{T}_i$, such that periodically all agents of the network are selected to update their commands, i.e. $\exists\, t^{'} : \forall t > 0, \bigcup_{i=0}^{t^{'}} \mathcal{T}_{t+i} = \mathcal{A}$. Then:

1) (*feasibility*) The overall system actuated by agents implementing Algorithm 1a never violates the constraints, i.e., $c(t) \in \mathcal{C}$ and (22) are satisfied for all pairs $(i, j) \in \mathcal{A} \times \mathcal{A}$ for all time steps $t \in \mathbb{Z}_+$.
2) (*stability*) The sequence of selected commands $g_i(t), t = 0, 1, \ldots$ is bounded for any arbitrary bounded reference sequence $r_i(t) \in \mathbb{R}^m$, for all i $\in \mathcal{A}$. BIBO stability of the overall system is guaranteed.
3) (*optimality*) for each agent $i \in \mathcal{A}$, at each decision time $t = k\tau, k \in \mathbb{Z}_+$, the minimizer of (11) uniquely exists and can be obtained by locally solving a convex optimization problem whose cost is monotonically decreasing whenever $r(t) \equiv [\ r^T_1, \ldots, r^T_N\ ]^T, \forall t$, with r_i a constant set-point.

Proof.

1) Under a global perspective, at each decision time $t \in \mathbb{Z}_+$, an admissible command $g(t)$ is applied to the overall network. By construction, the latter implies that the set-valued virtual predictions along the virtual time k satisfy

$$c(k, x(t), g(t)) \in \mathcal{C},\ \forall i \in \mathbb{Z}_+\,.$$

Then, the statement is proved by noticing that the following inclusion

$$c(t) = c(0, x(t), g(t)) \in \mathcal{C},$$

holds true at time instant t and by repeating the same argument for all $t \in \mathbb{Z}_+$.
2) from the previous item, it directly follows that the existence of a bounded $g_i(t)$, $i \in \mathcal{A}$ for all $t > 0$ is guaranteed by the existence of bounded $g_i(0)$. In fact, $g_i(0)$ is always a feasible solution at time 1. Moreover, because the internal controller guarantees local asymptotic stability, for a given bounded $g_i(t)$, the state evolution $x_i(\cdot)$ will also remain bounded. Then, BIBO stability of the overall system follows because the subsystems are dynamically decoupled.
3) The existence of an admissible solution for each agent at each decision time t follows from the fact that the previously applied reference $g_i(t) = g_i(t-1)$ is always an admissible, although not necessarily the optimal, solution for the prescribed problem at time t. As a consequence, the sequences of local costs $||g_i(t) - r_i||^2_{\Psi_i}$ are non increasing for any $i = 1, \ldots, N$ under constant set-points because it would change only if a "better" $g_i(t)$ is selected.

A. Link with DrPP

The proposed method is based on a solution of the drinking philosophers problem introduced by [23] that is derived from the well-known dining philosophers problem proposed by [24]. These problems capture the essence of conflict resolution, where multiple resources must be allocated to multiple processes. Given a set of processes and a set of resources, it is assumed that each resource can be used by at most one process at any given time. In our setting, processes and resources correspond to agents and tokens, respectively. More precisely, in **Algorithm 2**, τ_i represents the sequence of tokens related to the list of neighbors of agent i.

Within this particular context, a desired solution would have the following properties. (i) *Liveness*: in our setting liveness implies that each agent is eventually allowed to execute. (ii) *Fairness*: In multi agent setting, fairness indicates that there is no fixed priority order between agents. (iii) *Concurrency*: Any pair of processes must be allowed to execute at the same time, as long as they wish to access different resources. Analogously, no agent waits unnecessarily. The existing findings on the DrPP solution from [23] ensure the fulfillment of the mentioned properties. Therefore, it is reasonable to assert that even the proposed token-based enjoys this feature and a formal proof will be the object of a future study. On the other hand, several simulations (see next sub-section) have reported no evidence of possible deadlock situations.

B. Link with Graph Colorability

In [19], it has been proven the determination a sequence of non-overlapping turns in (2.6) is equivalent to solving the well-known graph coloring problem for Γ. For these reasons, Theorem 1 has motivated the investigation of the capability of the above introduced token-based protocol to find a minimal coloring configuration. In this respect, we present only a numerical analysis as this aspect goes beyond the main goal of this work. A more comprehensive and formal analysis will be developed in a future research work.

Specifically, the current objective is to observe the number of colors (turns) generated by the proposed token-based protocol. To achieve this, several simulations on different graph configurations have been conducted, and the results have been compared to the minimal coloring centralized solution. More precisely, within a set of 2800 randomly generated graphs denoted as Γ, the colors generated by the proposed method, represented as $\phi(\Gamma)$, have been compared with the *chromatic number* of each graph, denoted as $\hat{\phi}(\Gamma)$. For each graph, the probability of encountering a difference $\phi(\Gamma) - \hat{\phi}(\Gamma)$ equal

to a specific value X, i.e., $P(\phi(\Gamma) - \hat{\phi}(\Gamma) = X)$, has been evaluated and grouped by taking into account

1) the normalized Algebraic Connectivity λ_2 defined as the second smallest eigenvalue of the symmetric normalized Laplacian that, in our case, can be defined as $L := I - (D)^{-\frac{1}{2}} A (D)^{-\frac{1}{2}}$
2) the percentage ratio between the number of edges $|\mathcal{E}|$ and the maximum potential number of edges $\mathcal{E}_{max}(N) := (N(N-1)/2)$, i.e. $\mathcal{E}_{\%} := \frac{|\mathcal{E}|}{\mathcal{E}_{max}} \times 100$

Note that the minimum number of edges in a connected graph is $|\mathcal{E}| = N - 1$ [31]. Results are reported in Figures 7 and ar-

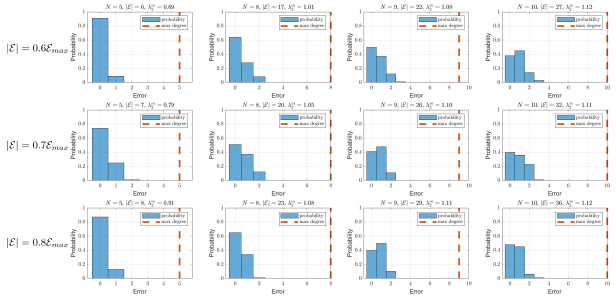


Fig. 7. Token based protocol performance analysis. Each plot shows the results computed on 100 experiments. λ_2^m is the average value of λ_2.

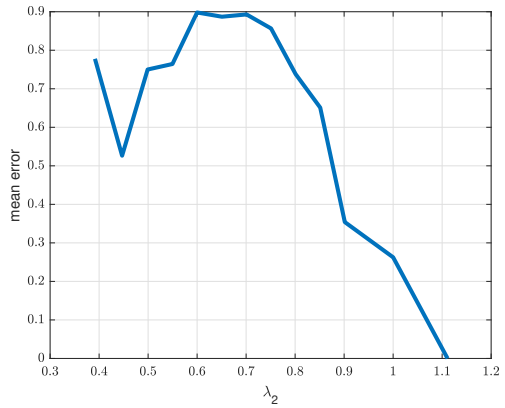


Fig. 8. Mean probability error v.s. λ_2.

ranged according to a grid pattern. Columns identify a constant number of nodes while rows are characterized by constant values of the parameter $|\mathcal{E}|$. For each test the number of nodes N, the number of edges $|\mathcal{E}|$ and the average value of λ_2 are reported. By considering that interesting practical scenarios may involve formations of vehicles characterized by a relatively small number of nodes and a high number of communication links, tests are performed for values of $N = (5, 8, 9, 10)$ and for each $|\mathcal{E}| = (60\%\mathcal{E}_{max}, 70\%\mathcal{E}_{max}, 80\%\mathcal{E}_{max})$. Hence, the communication graph of these types of formations is characterized by highly connected components with a small amount of nodes. Figure 7 shows that the proposed token heuristic is able to identify the optimal number of colors when applied to graphs with $|\mathcal{E}| >> N$ with high probability. For example, for graphs with $N = 10$ and $|\mathcal{E}| = 41$, the computed probability of finding the optimal number of colors is close to 0.8. It is interesting to notice that the maximum number of exceeding colors recorded during the tests is 4 $(N = 10, |\mathcal{E}| = 32)$ with a probability smaller than 0.01.

The previous results are further supported by Figure 8, which essentially depicts the progression of the mean error concerning the centralized optimal solution across different values of λ_2. The aim of this experiment is to examine the evolution of this error by systematically varying the number of edges. Specifically, we considered a set of $N = 10$ nodes and generated 2200 random graphs. Starting from 23 edges (equivalent to 50% of $\mathcal{E}_{max}$), we incrementally increased the number of edges by one unit up to 45, resulting in a complete graph. Graphs with identical λ_2 values were grouped, and the average error value was computed.

TABLE II
WORST CASE ANALYSIS.

N_n \ $\lvert\mathcal{E}\rvert$	$\mathbf{50\%}\mathcal{E}_{max}$	$\mathbf{60\%}\mathcal{E}_{max}$	$\mathbf{70\%}\mathcal{E}_{max}$	$\mathbf{80\%}\mathcal{E}_{max}$
5	(3,4,0)	(4,4,1)	(4,3,1)	(5,4,1)
6	(4,4,1)	(5,4,1)	(6,5,2)	(6,5,2)
7	(5,4,2)	(5,5,0)	(6,5,1)	(7,6,2)
8	(5,6,1)	(7,7,2)	(7,6,2)	(7,7,1)
9	(6,5,3)	(7,7,3)	(7,8,2)	(8,7,2)
10	(7,7,3)	(7,8,2)	(9,9,3)	(9,9,3)
15	(10,11,5)	(11,13,5)	(11,12,4)	(12,14,4)

A further analysis on worst case scenarios is reported in Table II. Each entry of the table is represented by the triplet $(\phi(\Gamma), \deg_i, \phi(\Gamma) - \hat{\phi}(\Gamma))$. The worst case analysis shows how the token-based protocol, when considering its graph coloring qualities, is able to find the minimum number of colors with a maximum error of 2 for graphs with less than 8 nodes. On the other hand, the number of exceeding colors increases with the number of nodes. In all the tests the number of colors computed by means of the token heuristic algorithm was always below the upper bound defined by the maximum degree of the graph *degree*$(\Gamma) + 1$ (dashed lines in Figure 7).

VIII. ILLUSTRATIVE EXAMPLES

In this section the main goal is to simulate, and therefore validate, the previously proposed method. In particular, the effectiveness of dynamic formations is demonstrated through three examples, involving both static and dynamic constraints. The vehicles are modeled as

$$\begin{cases} m\,\ddot{p}_i^x(t) + \beta\,\dot{p}_i^x(t) = T_{x,i}(t), \\ m\,\ddot{p}_i^y(t) + \beta\,\dot{p}_i^y(t) = T_{y,i}(t), \end{cases} \tag{33}$$

where $p_i^x(t)$ and $p_i^x(t)$ are the Cartesian coordinates, $T_{x,i}(t)$ and $T_{y,i}(t)$ are the manipulable inputs, the parameter $m = 10\,[kg]$ is the mass and $\beta = 1$ is the friction coefficient. The model (33) is pre-compensated by solving a $\mathcal{R}$-stabilizable control design problem and then discretized with sampling

time $T_c = 0.1\,[s]$. In each simulation, each vehicle i is required to satisfy the following local constraints

$$\begin{cases} ||T_{l,i}(t)|| \leq T_{max}\,[N], \\ ||\dot{p}_i^l(t)|| \leq v_{max}\,[\frac{m}{s}], \\ \text{with } l = \{x, y\} \text{ and } \forall t \in \mathbb{Z}_+. \end{cases} \tag{34}$$

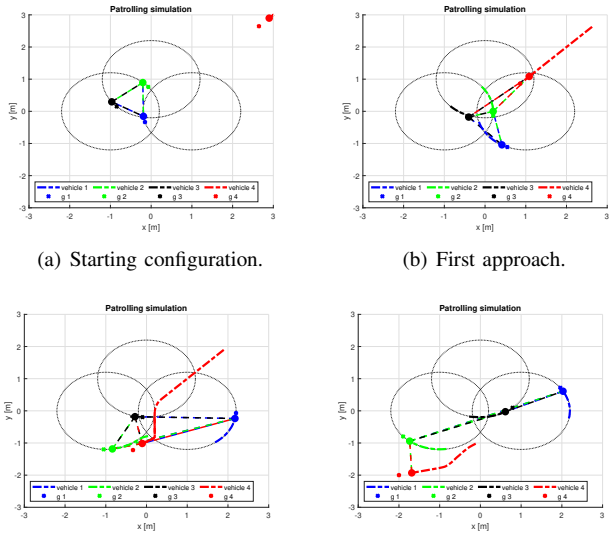
The algorithms were implemented in MATLAB ©, leveraging the YALMIP [32] and CoGoV toolboxes [38]. Simulations were performed on a workstation equipped with an 8-core AMD Ryzen 7 5800X3D (4.5 GHz) and 32 GB of DDR4-3200 memory.

A. Patroling

The first simulation scenario (depicted in Figure 9) is presented, where agent i_4 (red vehicle) is instructed to move through an area patrolled by a formation that consists of three agents, agent i_1 (black vehicle), agent i_2 (green vehicle) and agent i_3 (blue vehicle). The minimum admissible distance d_{min}, related to the *Collision Avoidance* constraints, is equal to $0.6\,[m]$ while the safe area radius is $R_{safe} = 0.7\,[m]$. Moreover, $R_1 = 2.4\,[m]$ and $R_2 = 2.8\,[m]$. The values of T_{max} and v_{max} are set to $50\,[N]$ and $0.5\,[\frac{m}{s}]$ respectively. Agent i and j belonging to the formation share proximity constraints of the form

$$||p_i(t) - p_j(t)||_\infty \leq d_{max} \tag{35}$$

where $d_{max} = 20\,[m]$ and have to move along a circular path. The different paths intersect in different points (Figure 9(a)).



(a) Starting configuration. (b) First approach.

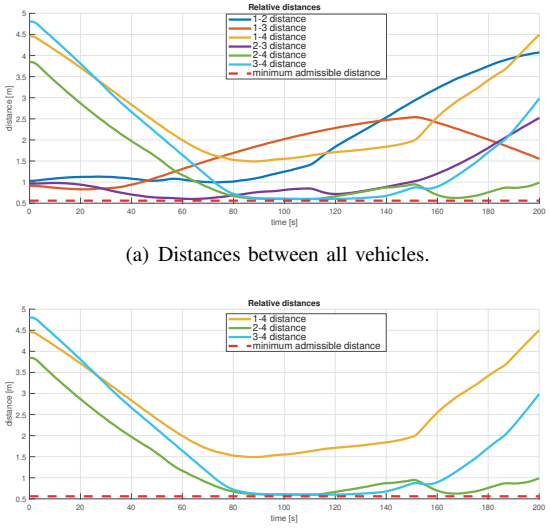
(c) Anti-collision maneuvers. (d) Final configuration.

Fig. 9. Patrolling simulation. Interactions between the formation and the outsider vehicle.

In order to avoid collisions within the formation, *Collision Avoidance* prescriptions are included from the beginning and are maintained active during the whole simulation. The existence of *Collision Avoidance* constraints is depicted in Figure 9 with two-colored dashed lines between the involved vehicles. During the simulation, another agent, i_4 (red colored vehicle), approaches the formation's operative area from the top right corner. Please notice that at the beginning of the simulation, agent i_4 does not belong to the formation, as no interactions are established. Whenever the distance between agent i_4 and another agent reaches the threshold defined by $R_1^\star$, a *plug-in* operation is requested and an interconnection is eventually established. Figure 9(b) shows the connections between agent i_4 and the formation.

An interesting configuration of agent i_4, agent i_1 and agent i_2 can be analyzed in Figure 9(c). The nominal straight path followed by agent i_4 becomes no longer feasible because it would violate the *Collision Avoidance* constraints shared with both agent i_1 and agent i_2. Agent i_4 then deviates from the original path and moves in a downward direction. Please notice that in Figure 9(c) the trajectory related to agent i_1 is no longer visible since such agent, in order to avoid collision with agent i_4, stands still for some time instants. Specifically, it cannot proceed along any path without violating constraints or without increasing its cost function value. Figure 9(d) shows an instant of the simulation where every vehicle is able to move freely.

To validate the proposed scheme, Figure 10 shows the relative distances recorded during the whole simulation. In particular, Figure 10(a) depicts the distances between all four agents. It is possible to notice that they never violate the *Collision Avoidance* constraints set at $d_{min} = 0.6\,[m]$. For the sake of clarity, the dynamic constraints involving agent i_4 are highlighted in Figure 10(b). The constraints are saturated at least once but they are never violated. Figure 11 shows



(a) Distances between all vehicles.

(b) Distances between agent i_4 and the vehicles belonging the formation.

Fig. 10. Patrolling simulation. Relative distances kept between agents.

the turn assigned to agent i_4 after each *PnP* operation. Each horizontal dashed line identifies a different turn. Turn membership of agent i_4 is determined by a token-based algorithm with respect to the active constraints with other agents. At the beginning of the simulation, agent i_4 has established no connection with the patrolling formation. Thus, because no turns are yet assigned to it, agent i_4 can compute its

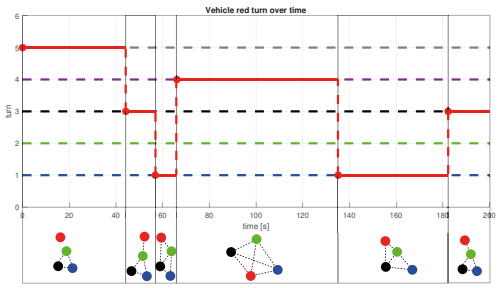


Fig. 11. Online turn membership computation for agent i_4 (red colored vehicle) and the existing formation using the turns determination token-based heuristic.

command at each sampling time t. When the relative distance between agent i_4 and agent i_2 becomes close to $R_1^\star$, agent i_4 sends a *plug-in* request to agent i_2 and a *PnP* operation is performed. Following the logic of the proposed algorithm, agent i_2 acquires the token of agent i_4. This implies that a turn is assigned to agent i_4, more specifically the same turn of agent i_1.

When a new *plug-in* request is sent from agent i_4 to agent i_1, after the *PnP* operation agent i_1 receives the token of agent i_4. As a result, agent i_4 now belongs to the same turn of agent i_3. When agent i_4 sends another *plug-in* request and performs a *PnP* operation with agent i_3, thus making the formation fully connected, a new turn (purple color) is assigned to agent i_4.

Before time instant $t = 140\,[s]$, the relative distance between agent i_4 and agent i_3 increases until it gets close to the disconnection distance R_2. Then a *plug-out* is requested and the connection between the agents is removed (with consequent local CGs re-configurations). Meanwhile, agent i_4 changes again its turn, belonging now to the same turn of agent i_3. When also the connection between agent i_4 and agent i_1 is removed, agent i_4 again belongs to the same turn of agent i_1. The maximum and average solution times required by the solver were $T_{max} = 0.00847\,[s]$ and $T_{avg} = 0.00547\,[s]$ respectively. Note that both of them are considerably shorter than the sampling time, T_c. A video of the complete simulation is available at the following link https://youtu.be/hEXJSNJQCns.

B. Collision between two formations

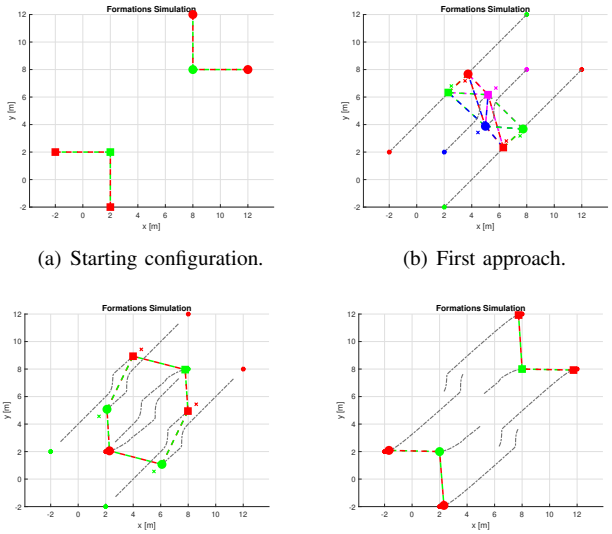
In the following simulations, the initial positions of two separate formations and their respective references are arranged in such a way to make vehicles collide differently, head-on and side collisions, making the introduction of the *Collision Avoidance* constraints crucial. The goal is, with each experiment, to highlight different aspects of the proposed solution. To be more precise, one scenario aims to study the distances maintained between agents, while the other shows how the network is colored by the suggested approach. Please note that agent i and agent j, belonging to the same formation, share proximity constraints (35), and thus cannot unplug.

Vehicles belonging to the two different formations are represented by different shapes: squares and circles. Furthermore, in each simulation, the computed admissible command g (represented as '$\times$') and the given reference r (represented as '$\circ$') are colored in the same way as the vehicle they are associated with. The parameters of the simulations are the following:

- v_{max} set to $0.25\,[\frac{m}{s}]$ in the first experiment and $0.5\,[\frac{m}{s}]$ in the second experiment;
- $R_1 = 4\,[m]$;
- $R_2 = 6\,[m]$;
- $R_{safe} = 1\,[m]$;
- $d_{max} = 5\,[m]$;
- $d_{min} = R_{safe} = 1\,[m]$.

Note that the value $R_1 = 4\,[m]$ is admissible. In fact, the equation (23) is verified by choosing $R_1^\star = 4\,[m]$ and $\varepsilon = 1$.

1) Head-on Collision: In this scenario, two formations, sharing proximity constraints, are instructed to 'swap' positions. Vehicles belonging to different formations collide head-on in this manner as they move in opposing directions. Figure



(a) Starting configuration. (b) First approach.

(c) Anti-collision maneuvers. (d) Final configuration.

Fig. 12. Head-on collision simulation.

12 shows the positions of vehicles and their interactions at different times during the simulation. Vehicles from the two formations are initially spatially disposed to form a triangle (Figure 12(a)), and vehicles belonging to separate formations do not interact.

As the simulation progresses, Figure 12(b) shows the new interactions that arise to avoid collisions. After some *Collision Avoidance* maneuvers, constraints are gradually removed when they are no longer needed, as shown in Figure 12(c) and Figure 12(d).

It is interesting to note that, although the constraints between vehicles of the same formation only preserve connectivity, the triangular shaped geometrical spatial disposition of the formations is preserved even after the *Collision Avoidance* maneuvers. The turn membership procedure is analyzed in more detail in Figure 13, which represents the exact configuration of the graph related to the time instants depicted in Figure 12. At the beginning of the simulation, since no interaction is present between the two formations, the entire graph is not connected. Since the vehicle pairs $i_2 - i_3$ and $i_4 - i_6$ do

not share constraints, only two turns (colors) are needed to partition the networks into non-neighboring nodes.

As more connections are established, more colors are needed to properly arrange the vehicles into turns. A trend consistent with previous maximum and average solution times was observed in this scenario, with $T_{max} = 0.0212\,[s]$ and $T_{avg} = 0.0072\,[s]$ respectively. As connections are gradually removed, the number of turns decreases. In this way, more vehicles can compute their respective commands in parallel. Please note that the number of turns (colors) in this simulation is always minimal. A complete video of the simulation is available on line at the link https://youtu.be/VTilInwcSj0.

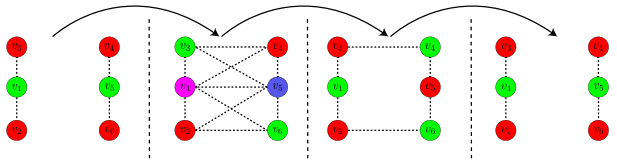
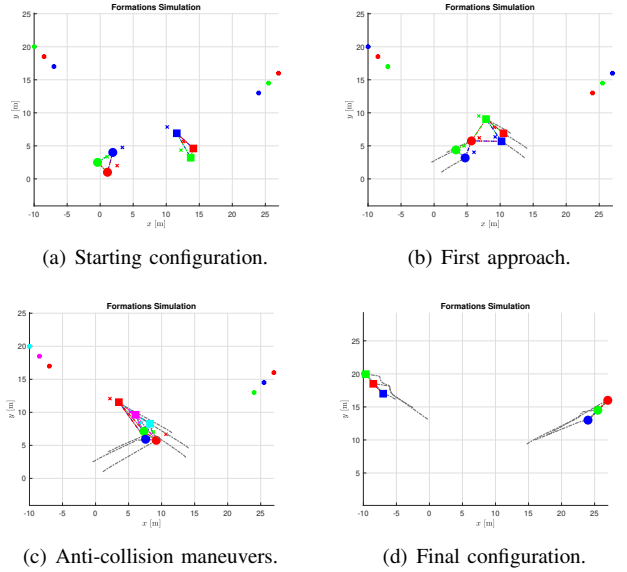


Fig. 13. Turns assigned to each agent in different instants of the simulation using **Algorithm 2**.

2) Side Collision: In the last proposed simulation, it is assumed that each formation is executing a specific task, and that its path intersects with the path of the other formation for a certain length of time. It is crucial for each formation to reach given aligned references while ensuring *Collision Avoidance*.

More specifically, the formation initially located on the left side, without the supervision architecture, would collide with the left side of the formation initially located on the right (Figure 14(a)). Although this scenario is similar to the previous one, in this case vehicles belonging to the same formation are closer, so the number of shared constraints associated with each vehicle is higher, resulting in a more complex optimization problem. Figure 14 shows the trajectories recorded during the simulation. Figure 14(b) shows the



(a) Starting configuration. (b) First approach.

(c) Anti-collision maneuvers. (d) Final configuration.

Fig. 14. Side collision simulation.

initial interactions between the two formations. The plug-in operations are requested because the distances between vehicles of different formations fall below the predefined threshold R_1. To ensure the feasibility of the constraints, *Collision Avoidance* maneuvers are performed as shown in Figure 14(d), and finally the references are safely reached. The evolution of

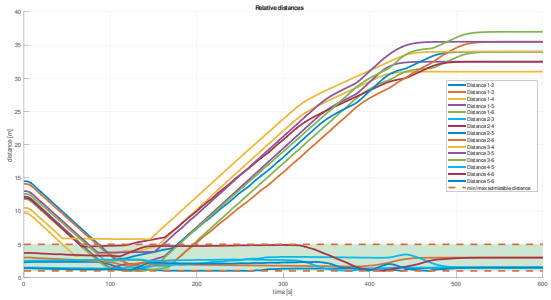


Fig. 15. Path crossing simulation. Distances recorded over time.

the relative distances between vehicles is shown in Figure 15. To this extent, it is possible to provide a detailed analysis of the feasibility of the *Collision Avoidance* constraints. The measured distances between vehicles of the same formation are enclosed in the green region. More precisely, these distances are kept between $1\,[m]$ and $5\,[m]$ (the minimum and maximum distances respectively). The other reported distances are associated to distances of vehicles belonging to different formations.

The collision between the two formations is avoided after $t = 100\,[s]$. Although the vehicles are dangerously close to collision (constraints are saturated), the distances never fall below the minimum allowable distance d_{min}. When the relative distance between agents belonging to different formations grows larger than the predefined threshold R_2, plug-out operations are requested. A second collision, this time between vehicles of the same formation, is avoided at the end of the simulation. In this scenario as well the time required by the solver to find optimal solutions remains consistently below the sampling time, with $T_{max} = 0.0241\,[s]$ and $T_{avg} = 0.0052\,[s]$ respectively. A video of the complete simulation is available online at the link https://youtu.be/gnIZPlREKmY.

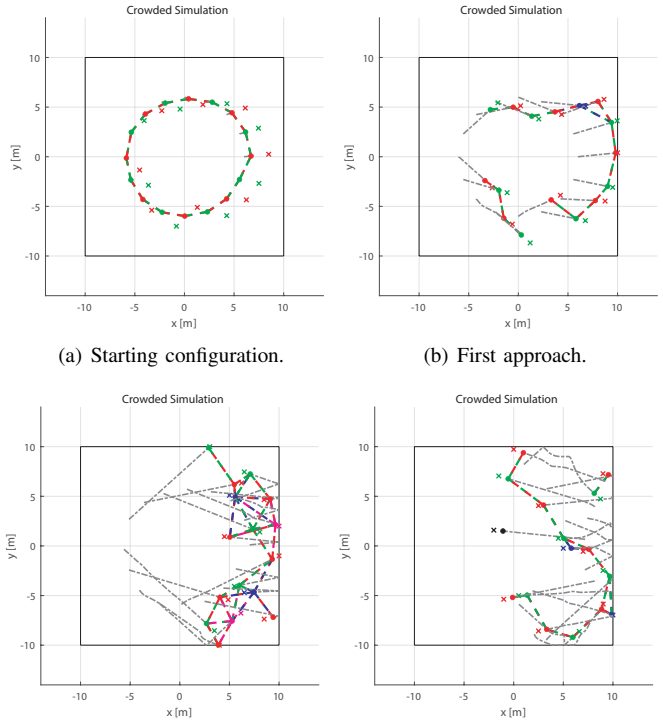
C. Crowded constrained environment

To further demonstrate the effectiveness of the proposed strategy, this section presents a simulation involving an increased number of vehicles operating within a constrained environment. Specifically, $N = 16$ vehicles, initially arranged as depicted in Figure 16(a), are required to move within a confined square-shaped area.The vehicle are subject to constraints (34), and to additional constraints to confine them within the area of interest, i.e.

$$||p_i^l(t)|| \leq p_{max}\,[m], \tag{36}$$

with $p_{max} = 10\,[m]$ and $l = \{x, y\}$. Vehicles that reach the boundaries of the environment exhibit a bounce-back behavior, similar to that observed in a game of pool. This mechanism

increases the likelihood of random collisions between vehicles, thereby enhancing the complexity of the simulation. The simulation time is $t_{\text{fin}} = 48.5\,[s]$. As the simulation progresses and



(a) Starting configuration. (b) First approach.

(c) Anti-collision maneuvers. (d) Final configuration.

Fig. 16. Constrained environment simulation.

the vehicles begin to move, the *PnP* procedures cause some connections to be removed, resulting in the creation of smaller formations, as shown in Figure 16(b). When certain vehicles reach the right boundary of the environment and bounce back, they may encounter vehicles incoming from the opposite direction. In such cases, using the proposed approach, new connections are dynamically established to enforce collision avoidance, as depicted in Figure 16(c). Once collisions are successfully avoided, the billiard-like behavior causes vehicles to move away from one another, leading to a significant decrease in inter-vehicle connectivity, as shown in Figure 16(d). It is interesting to note that the proposed simulation highlights an edge case in which formations may consist of a single vehicle. For a more in-depth analysis, Figure 17 presents the inter-vehicle distances for those pairs that come dangerously close to violating the collision avoidance constraints. Thanks to the supervision architecture, vehicles perform anti-collision maneuvers to maintain at least a minimum distance of $d_{\min}$ between each other. As a result, the collision avoidance constraints are always satisfied and, in the worst-case scenario, they are saturated. The maximum time recorded to find feasible solution in this scenario is $T_{max} = 0.0136$ while the average time is $T_{avg} = 0.0092$. A video of the simulation is available at https://youtu.be/ehkCq0AeeYw.

IX. Conclusions

In this paper, the problem of guaranteeing *Collision Avoidance* prescriptions for a dynamic formation of agents moving

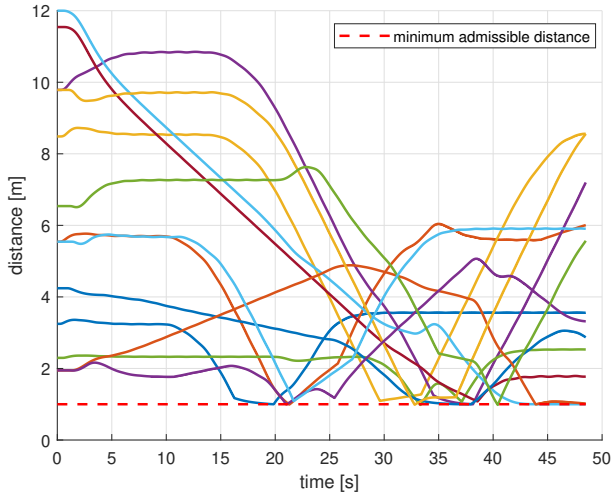


Fig. 17. Distance between pair of vehicles dangerously close to collision.

on two dimensional space has been addressed. The main effort of this work was to extend the solution presented in [21] to deal with dynamical inclusion and removal of non-convex *Collision Avoidance* constraints. Furthermore, a token heuristic was proposed to arrange agents into turns by solving the turn determination problem (graph coloring problem) in a distributed fashion and the performances of the proposed heuristic have been extensively analyzed. The resulting dynamic distributed supervision scheme was presented and its behavior illustrated in several simulated experiments. Future works may involve the application of the proposed approach to nonlinear and, in particular, non-holonomic systems.

Acknowledgment

This research is based upon work partially supported by the industrial research project *PON ARS01_00682*, entitled “ARES - Autonomous Robotic for the Extended Ship”, granted by the Italian Ministry of Economic Development (MISE) and European Union and the research project - ID:20225MESZB “Resilient Optimization in Distributed Cyber-Physical Control Problems subject to Adversarial Behaviour” granted by the Italian Ministry of University and Research (MUR), within the PRIN 2022 program, and European Union - Next Generation EU.

References

[1] P. Molnar and J. Starke, “Control of distributed autonomous robotic systems using principles of pattern formation in nature and pedestrian behavior,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, pp. 433-435, 2001.

[2] R. Cepeda-Gomez and L. F. Perico, “Formation Control of Nonholonomic Vehicles Under Time Delayed Communications,” *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 819-826, 2015.

[3] C. Wang, W. Cai, J. Lu, X. Ding, and J. Yang, “Design, Modeling, Control, and Experiments for Multiple AUVs Formation,” *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 2776-2787, 2022.

[4] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control: Collective group behavior through local interaction," *IEEE Control Syst. Mag.*, vol. 27, no. 2, pp. 71–82, 2007.

[5] R. Wang, X. Dong, Q. Li, and Z. Ren, "Distributed Time-Varying Formation Control for Linear Swarm Systems With Switching Topologies Using an Adaptive Output-Feedback Approach," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 12, pp. 2664-2675, 2019.

[6] G. Wang, X. Wang, and S. Li, "A Guidance Module Based Formation Control Scheme for Multi-Mobile Robot Systems With Collision Avoidance," *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 1, pp. 382-393, 2024.

[7] X. Dong, B. Yu, Z. Shi, and Y. Zhong, "Time-varying formation control for unmanned aerial vehicles: Theories and applications," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 1, pp. 340–348, 2015.

[8] J. Cui, Y. Zhang, S. Ma, Y. Yi, J. Xin and D. Liu, "Path planning algorithms for power transmission line inspection using unmanned aerial vehicles," In Proceedings of the 2017 29th Chinese Control And Decision Conference (CCDC), 28-30 May, Chongqing, China, pp. 2304-2309, 2017.

[9] T. Wakabayashi, Y. Suzuki, and S. Suzuki, "Dynamic obstacle avoidance for multi-rotor UAV using chance-constraints based on obstacle velocity". *Robotics and Autonomous Systems*, vol. 160, pp. 104320, 2023.

[10] B. Wang, J. Wang, B. Zhang, and X. Li, "Global cooperative control framework for multiagent systems subject to actuator saturation with industrial applications," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 7, pp. 1270–1283, 2017.

[11] Q. Zou, X. Du, Y. Liu, H. Chen, Y. Wang, and J. Yu, "Dynamic Path Planning and Motion Control of Microrobotic Swarms for Mobile Target Tracking," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 4, pp. 2454-2468, 2023.

[12] D. Sun and J.K. Mills, "Adaptive synchronized control for coordination of multirobot assembly tasks," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, pp. 498–510, 2002.

[13] W. Ren and N. Sorensen, "Distributed coordination architecture for multi-robot formation control," *Robotics and Autonomous Systems*, vol. 56, no. 4, pp. 324–333, 2008.

[14] T. Arai, E. Pagello, and L.E. Parker, "Advances in multi-robot systems," *IEEE Transactions on robotics and automation*, vol. 18, no. 5, pp. 655–661, 2002.

[15] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," *Springer Tracts in Advanced Robotics*, pp. 649–666, 2016.

[16] S. Ahmed, B. Qiu, F. Ahmad, C. Kong, and H. Xin, "A State-of-the-Art Analysis of Obstacle Avoidance Methods from the Perspective of an Agricultural Sprayer UAV's Operation Scenario," *Agronomy*, vol. 11, no. 6, 2021.

[17] G. Ferrari-Trecate, L. Galbusera, M. P. E. Marciandi, and R. Scattolini, "Model predictive control schemes for consensus in multi-agent systems with single and double integrator dynamics," *IEEE Trans. Autom. Control*, vol. 54, no. 11, pp. 2560–2572, 2009.

[18] B. Ding, L. Ge, H. Pan, and P. Wang, "Distributed MPC for Tracking and Formation of Homogeneous Multi-agent System with Time-Varying Communication Topology," *Asian Journal of Control*, vol. 18, pp. 1030–1041, 2016.

[19] A. Casavola, E. Garone, and F. Tedesco, "A Distributed Command Governor Based on Graph Colorability Theory," *Int. Journal of Robust and Nonlinear Control*, vol. 28, pp. 3056–3072, 2017.

[20] F. Tedesco, D. M. Raimondo, and A. Casavola, "Collision avoidance command governor for multi-vehicle unmanned systems," *Int. J. Robust. Nonlinear Control*, pp. 2309–2330, 2014.

[21] F. Tedesco, S. Sarkar, and A. Casavola, "Turn-Based Supervision Architectures for Dynamic Networks involving Plug-and-Play Operations," *IFAC-PapersOnLine*, vol. 52, no. 3, pp. 90-95, 2019.

[22] F. Tedesco and A. Casavola, "Distributed Supervision Strategies for Cyber-Physical Systems with Varying Network Topology," *IEEE Transactions on Automatic Control*, pp. 1-16, 2024.

[23] K. M. Chandy and J. Misra, "The drinking philosophers problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 6, no. 4, pp. 632–646, 1984.

[24] E. W. Dijkstra, "Hierarchical ordering of sequential processes," *The origin of concurrent programming*, Springer, pp. 198–227, 1971.

[25] R.M. Karp, "Reducibility among combinatorial problems," *In: Miller R.E., Thatcher J.W., Bohlinger J.D. eds. Complexity of Computer Computations. Boston, MA: Springer*, pp. 85-103, 1972.

[26] L. Barenboim and M. Elkin, "Distributed Graph Coloring: Fundamentals and Recent Developments," *Synthesis Lectures on Distributed Computing Theory*, vol. 4, no. 1, pp. 1-171, 2013.

[27] T. Schouwenaars, J. How, and E. Feron, "Receding horizon path planning with implicit safety guarantees," *Am. Control Conference*, The Bahamas, pp. 5576–5581, 2004.

[28] F. Asano, "Stability analysis of underactuated bipedal gait using linearized model," *11th IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia, 2011.

[29] A. Richards and P.G. How, "Aircraft Trajectory Planning With Collision Avoidance Using Mixed Integer Linear Programming," *IEEE American Control Conference*, Anchorage, AK, USA, 2002.

[30] A. Casavola, E. Garone, and F. Tedesco, "A distributed command governor based on graph colorability theory," *International Journal of Robust and Nonlinear Control*, pp. 3056–3072, 2018.

[31] X. Li, M. Z. Q. Chen, H. Su, and C. Li, "Distributed Bounds on the Algebraic Connectivity of Graphs With Application to Agent Networks," *IEEE Transactions on Cybernetics*, vol. 47, no. 8, pp. 2121-2131, 2017.

[32] J. Lofberg, "YALMIP: A toolbox for modeling and op-

timization in MATLAB," *in Proc. IEEE Int. Conf. Robot. Autom.*, Taipei, Taiwan, Aug. 2004, pp. 284–289.

[33] Gurobi Optimization, LLC. (2025). *Gurobi Optimizer Reference Manual*. [Online]. Available: https://www.gurobi.com.

[34] A. Casavola, A. El Qemmah, F. Tedesco, and F. A. Torchiaro, "Coordination of Marine Multi-Vehicle Autonomous Systems via the Distributed Command Governor Approach," *in Proc. IEEE Oceans*, Limerick, Ireland, pp. 1-8, 2023.

[35] A. Casavola, V. D'Angelo, A. E. Qemmah, G. Gagliardi, F. Tedesco and F. A. Torchiaro, "Preserving Agents Connectivity Amidst Static Obstacles: A Distributed Predictive Supervision Approach within Robot Operating System," *IEEE Transactions on Intelligent Vehicles*, 2024.

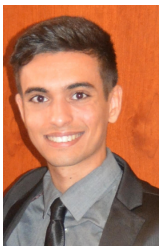
[36] A. Koubâa, *Robot Operating System (ROS): The complete reference. Vol. 1*, Springer, New York, 2017.

[37] S. Diamond and S. Boyd, "CVXPY: A Python-Embedded Modeling Language for Convex Optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1-5, 2016.

[38] A. Casavola, V. D'Angelo, A. El Qemmah, G. Gagliardi, F. Tedesco and F.A. Torchiaro, "A Matlab-Based Toolbox for Supervising Multi-Vehicle Autonomous Systems," *IEEE Access*, vol. 12, pp. 127051-127064, 2024.

Alessandro Casavola received the Ph.D degree in Systems Engineering from the University of Bologna, Italy, in 1990. He has been with the Department of Systems and Computer Engineering of the University of Florence from 1996 to 1998 as a Researcher. Since 1998 he is with the Department of Computer Science, Modeling, Electronics and Systems Engineering of the University of Calabria: as an Associate Professor first and as Full Professor since 2005. His current research interests include constrained predictive control, control under constraints, control reconfiguration for fault tolerant systems and supervision approaches over data networks. He has been the Program Chair of IEEE ISIC 2014 Symposium (within the IEEE Multi Conference on System and Control (IEEE MSC 2014), held in NICE, France on October 2014 and the Program Co-Chair of SysTol 2019, held in Casablanca, Morocco, Sep. 18-20, 2019. He is a Subject Editor of the International Journal of Adaptive Control and Signal Processing since 2009 and served as an Associate Editor of the IEEE Control Systems Letters from 2018 to 2022. He is the co-recipient of the Journal of Process Control Best Survey Paper Award 2023 and the co-recipient of the 2023 IEEE Vehicle Power and Propulsion Conference (VPPC 2023) Best Paper Award.

Vincenzo D'Angelo received the B.S. degree in computer engineering in 2015 and the M.S. degree in automation engineering from University of Calabria, Rende, Italy, in 2018. From 2019 to 2022, he worked as Automation and Software Engineer in a spin-off of University of Calabria, and in the same period he was Contract Lecturer for the course of Mechatronics and Mobile Robotics Lab at the University of Calabria, Rende, Italy. Currently he works in the industrial sector and he continues to collaborate with the Department of Computer Engineering, Modeling, Electronics and System (DIMES) of University of Calabria as external collaborator. His research interest includes the development of robotics, IoT and embedded systems, with main focus on underwater and maritime applications.



Ayman El Qemmah received the B.S. degree in Computer Engineering and the M.S. degree in Automation Engineering from the University of Calabria, Rende, Italy, in 2017 and 2020, respectively. He is currently pursuing the Ph.D. degree in Information and Communication Technologies at University of Calabria, Rende, Italy. In 2023 he was a Visiting Scholar Research at the Washington University in St. Louis, Saint Louis, MO, USA. His current research interests include formation control, supervision approaches, constrained predictive control, data-driven supervision, autonomous vehicles and overactuated marine surface vehicles.

Gianfranco Gagliardi received the master's degree in automation engineering and the Ph.D. degree in systems and computer engineering from the University of Calabria, Italy, in 2007 and 2011, respectively. Since 2020, he has been an Assistant Professor with the University of Calabria. He is currently an Active Researcher with publications in different journals and international conferences. He also had an active role in many research projects both from academia and the private sector. His research interests include fault detection and isolation strategies, real-time embedded systems, and DSP-based computer vision solutions. He is co-recipient of the 2023 IEEE Vehicle Power and Propulsion Conference (VPPC) Best Paper Award.

Francesco Tedesco received his Master Degree in Automation Engineering in 2008 and his Ph.D. in Systems and Computer Engineering in 2012 from the University of Calabria, Italy. In 2010 he was a visiting Scholar Research at the ETH Zurich (Switzerland) and in 2014 he covered the same position at the Imperial College of London (United Kingdom). From 2012 to 2017 he was an Assistant Researcher at the DIMES department of the University of Calabria where, since 2018, he served first as Assistant Professor and then as Associate Professor since 2022. He is a co-recipient of the Best Paper Award at the IEEE-CoDIT 2019 Conference, Paris, France. His main research interests involve Supervision Approaches over Data Network, Model Predictive Control, Automotive Applications and Control of Cyber-Physical Systems. Since 2022 he is Senior Member of IEEE. He currently serves as an Associate Editor for IET Control Theory & Applications, IEEE Transactions on Automation Science and Engineering (TASE) and IEEE Open Journal of Control Systems. In recognition of his editorial contributions, he received the Outstanding Associate Editor Award for IEEE TASE in 2024. He is also an active member of the IFAC Technical Committee TC 6.4 on Fault Detection, Supervision, and Safety of Technical Processes (SAFEPROCESS), where he contributes to the working group on "Safety and Security of Cyber-Physical Systems.

Franco Angelo Torchiaro received the B.S. degree in computer science and the M.S. degree in automation engineering from the University of Calabria, Italy, in 2017 and 2020, respectively, where he is currently pursuing the Ph.D. degree with the DIMES, Information and Communication Technology (ICT) Programme. His research interests include design, modeling, and performance evaluation of networks and distributed systems, with a focus on underwater sensor networks (USNs) and unmanned surface vehicle formations. Recently, his research has focused primarily on underwater communications and security in USNs, actively contributing to the development of an academic underwater acoustic modem, combining trust models, and sensor selection techniques to detect and mitigate the effects of malicious nodes.