Decentralized task allocation for redundant multi-robot systems: an iterative consensus approach

Lorenzo Govoni and Andrea Cristofaro

Abstract— **An overactuated heterogeneous multi-robot system is considered, this being characterized by both a redundancy of the number of agents with respect to the tasks to be performed and input redundancy for each individual agent. We propose an algorithm that, based on local information only, can simultaneously assign the tasks, consistently with the different nature of each robot, and allocate the control efforts while satisfying the constraints on the actuators. The performances of the proposed method have been analysed by means of a simulation study, considering different scenarios and illustrating also the resilience of the approach with respect to faults.**

I. INTRODUCTION

Multi-robot systems (MRS) are deployed to collectively perform tasks that a single agent may not achieve by itself. One fundamental challenge in MRS lies in determining which robot should execute which task to cooperatively achieve the global goal, known as *task allocation* [1], [2]. This problem involves considering the different nature of each agent and/or the constraints associated with the task execution. Optimal task assignment policies seek to maximize overall system performance [3]. However, scalability is a major concern in task allocation frameworks, particularly when the assignment is managed centrally. The amount of tasks and robots may penalize the computational efficiency of the algorithm. Methods like the one proposed in [4], offer a solution by employing market-based decision strategy [5], using the consensus protocol for resolving possible conflicts and reducing scalability issues.

Individual robots forming the MRS may possess more actuators than the ones strictly needed or meeting the requirements of a given application [6]. This redundancy presents an opportunity to address the so-called *control allocation* problem. The backbone of the control allocation setup is designing a control input $u \in \mathcal{U}$ to generate a desired virtual control $v = B(x,t)u$ jointly from the actuators at any time t. This characteristic enables the encoding of second objectives, typically chosen from an operational perspective, like minimization of power consumption or fault-tolerant criteria [7], [8].

In this paper, we extend our previous work on a fault-tolerant task allocation framework for overactuated multi-robot systems [9] to a fully decentralized setting, achieved through the integration of a consensus protocol. The two allocation architectures can be defined in terms of optimization problems, whose requirements are independent one from each other, since they work at two different control levels. The task allocation provides the desired virtual controls encoding the task execution performances, by means of quadratic control barrier functions (CBFs) with cross-terms for robots with second-order dynamics. Then, the outcomes of the task allocation are fed to the control allocation framework, mapping them into the actual control inputs and taking care of the constraints on the actuators of the agents. The consensus protocols help the robot to become aware of the actual capabilities of all the other agents and to detect the occurrence of possible faults in the system, thus enabling the robots to perform a decentralized and *resilient* task allocation. In conclusion, thanks to the decoupled behaviour of the two architectures, the overall framework results in a series of cascade schemes, one for each robot of the MRS.

The remainder of the paper is organized as follows. Section II describes the task allocation architecture, which is then further generalized in Section III where the cascade with the control allocation setup is analyzed. Simulation results, obtained using Matlab and CoppeliaSim, are reported in Section IV, while final conclusions are given in Section V.

Notation: $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ represents an undirected graph, where $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$ is the node set with cardinality $|\mathcal{V}| = N$, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, is the edge set, comprising unordered pairs of nodes. $\mathcal{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix, a symmetric matrix with entries $a_{ij} = 1$ for $(v_i, v_j) \in \mathcal{E}$ and $a_{ij} = 0$ otherwise. $\mathcal{D} \in \mathbb{R}^{N \times N}$ is the degree matrix, and $\mathcal{L} = \mathcal{D} - \mathcal{A}$ denotes the Laplacian matrix associated with $\mathcal{G}$.

II. CONSENSUS-BASED TASK ALLOCATION

Consider a generic linear control system in the state space representation

$$\begin{aligned} \dot{x} &= Ax + B_u u \\ y &= Cx + Du \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the input and $y \in \mathbb{R}^r$ is the output. In a multi-task multi-robot setting, each task t_m, with $m \in \{1, \dots, n_t\}$, can be described by a continuously differentiable positive definite cost $J_m : \mathbb{R}^n \to \mathbb{R}^+$ which is a function of the robot state x. The work done in [6] has proved that the execution of the task can be cast into designing a control that minimizes $J_m(x)$, where the state x and the control u are coupled through a dynamics like (1). Thanks to the extended set-based tasks formulated in [10], it is possible to encode a large variety of tasks by means of a set, which is to be made either forward invariant, i.e. safe, or asymptotically stable, or both, using a suitable control barrier function (CBF) $h_m(x)$ [11]. In particular, the

The authors are with the Department of Computer, Control and Management Engineering, Sapienza University of Rome, Via Ariosto 25, 00185 Rome, Italy. Corresponding author: L. Govoni, govoni@diag.uniroma1.it

accomplishment of the task t_m is translated into the inclusion $x \in \{x \in \mathbb{R}^n : h_m(x) = -J_m(x) \geq 0\}$. Building upon such CBF-based setting, the formal definition of the problem to be addressed in this paper is given in the following statement.

Problem 1: Design a task allocation framework able to optimally assign the tasks to a team of robots according to the different capabilities of each single agent, such that: $i)$ is resilient with respect to faults, e.g. changes in the robot capabilities, and $ii)$ the architecture is fully decentralized, i.e., both the allocation and the fault detection must be solved using only local information.

A. *Centralized task allocation algorithm*

Global task allocation specifications, along with the local constraints that each robot must satisfy, can be encoded through the *assignment* matrix $\alpha \in \{0,1\}^{n_t \times n_r}$ [3], where n_r stands for the number of robots in the system. It is assumed that, at any instant of time at most one element of $\alpha_{m,-}$[1] can be non-zero, due to the fact that each robot can be assigned, at most, to one task. The optimization-based formulation for the task allocation is the following:

$$\underset{\alpha}{\text{minimize}} \quad \sum_{i=1}^{n_r} \left(c||\Pi_i \alpha_{-,i}||^2 \right) \tag{2a}$$

$$\text{subject to} \quad \mathbb{1}_{n_t}^T \alpha_{-,i} \leq 1 \tag{2b}$$

$$F\alpha_{m,-}^T \geq T_{m,-}^T \tag{2c}$$

$$n_{r,m,min} \leq \mathbb{1}_{n_r}^T \alpha_{m,-}^T \leq n_{r,m,max} \tag{2d}$$

where all the constraints must be satisfied $\forall i \in \{1 \dots n_r\}$ and $\forall m \in \{1 \dots n_t\}$. The constant $c \in \mathbb{R}$ is an optimization parameter, while $n_{r,m,min}$ and $n_{r,m,max}$ encode respectively the minimum and maximum number of robots required by the task t_m. The matrices F and T encode the mappings that link the capabilities each task requires with the heterogeneity of each robot accordingly to the hypergraph like the one in Figure 1 (see [3] for further details on the computation of the mappings). Furthermore, in the cost function (2a), the matrix $\Pi_i = I_{n_t} - S_i S_i^\dagger$ is used to penalize bad allocations of tasks. The matrix S_i denotes the specialization matrix of robot i and its zero entries correspond to the tasks the robot i has no specialization to perform, yielding in a non-zero cost in (2a) through Π_i. The problem (2) is solved point-wise in time and so it can be integrated along with online updates to the specializations and capabilities of the robots, fulfilling item $i)$ of Problem 1.

Once the task allocation has been achieved, the robot i extracts the corresponding column i of the matrix α and computes the control solving the following QP problem:

$$\underset{u_i \in \mathbb{R}^m, \delta_i}{\text{minimize}} \quad ||u_i||^2 + l||\delta_i||_{S_i}^2 \tag{3a}$$

$$\text{subject to} \quad L_f h_m(x) + L_g h_m(x) u_i \geq -\gamma(h_m(x)) - \delta_{im} \tag{3b}$$

$$\Theta\delta_i + \Phi\alpha_{-,i} \leq \Psi \tag{3c}$$

$$||\delta_i||_\infty \leq \delta_{max} \tag{3d}$$

where $l \in \mathbb{R}$ and $\delta_{max} \in \mathbb{R}$ are optimization parameters and Θ, Φ and Ψ in (3c) are coefficient matrices that encode the prioritization of the tasks based on the assignment matrix α. The cost function in (3a) is made of two terms. The first concerns the norm of the control effort, making the algorithm compatible with long-duration autonomy applications. The second ensures the execution of the assigned task t_m while fulfilling constraint (3b). Among the n_t constraints collected in (3b), the one corresponding to t_m experiences the least relaxation, yielding in the smallest δ_m.

While the architecture (3) has been shown to be quite efficient [3], [9], it relies on the presence of a central authority in charge of allocating the tasks. Next we present a decentralized version of the same architecture, thus avoiding the need for a centralized task allocator.

[1]The symbols $X_{i,-}$ and $X_{-,j}$ refer respectively to the ith row and the jth column of the matrix X.

B. *Decentralized approach*

In a decentralized context, the agents of a multi-robot system communicate through a directed graph $\mathcal{G}$, assumed here to be undirected for the sake of simplicity, so that each robot can perform the task allocation based on local information. Let us assume that the matrix T is known by each robot, since it is an intrinsic property of the multitask scenario, whereas they do not know the mapping A, from which we obtain the mappings F and S, useful for solving (2). For this purpose, each robot is endowed with an extra state $x_{feat,i} \in \{0,1\}^{n_f \cdot n_r}$ in which it stores a vector representation of the matrix A, effectively encoding the estimated features of the other agents, updated by following the consensus-based procedure described in Algorithm 1.

Algorithm 1 Consensus on the mapping A

```
Inputs:
Adjacency matrix $\mathcal{A}$
Number of robots $n_r$ and number of features $n_f$
Vector state $x_{feat,i} \in \{0,1\}^{n_f \cdot n_r}$, $i = \{1, \dots, n_r\}$
1: for each robot $i$ do
2:     Initialize $x_{feat,i}$ with local information
3:     Do the consensus on $x_{feat,i}$                    ▷ (4)
4:     Normalize the non-zero component of $x_{feat,i}$
5:     Construct mapping $A_i$
6: end for
```

Furthermore, robot redundancy renders the system robust with respect to faults and, in a decentralized context, the overall fault awareness must be handled using only local information. Assuming that robots are able to perform the detection of faults occurring on their own actuators or features, let us define a binary vector $\beta^i \in \{0,1\}^{n_r}$ for each robot. This vector stores the faulty information within the

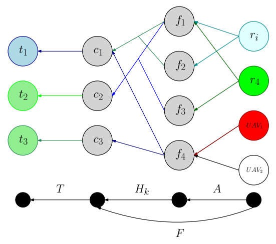


Fig. 1: Heterogeneity characterization of a team of robots. For graphical reasons, the robots that share the features f_1 and f_2 are represented all in one node r_i, with $i = 1, 2, 3, 6$.

system, with the value 0 corresponding to nominal conditions and 1 to the presence of a fault. Algorithm 2 provides a consensus-based fault detection procedure for the multi-robot system [12].

Algorithm 2 Consensus-based fault detection

```
Inputs:
Adjacency matrix $\mathcal{A}$
Number of robots $n_r$
Binary vector state $\beta^i \in \{0,1\}^{n_r}$, $i = \{1,\dots,n_r\}$
1: for each robot $i$ do
2:     Initialize $\beta^i$ with its own information
3:     Do the consensus on $\beta^i$                              ▷ (4)
4:     Normalize $\beta^i$
5:     if $\beta^i_j = 1$ for $j = \{1,\dots,n_r\}$ then
6:         Fault detected
7:     end if
8: end for
```

The joint use of Algorithm 1 and Algorithm 2, along with the task allocation procedure (2), enables the multi-robot system to solve item $ii)$ of Problem 1.

In conclusion, each robot composing the team has an overall state defined as $q_i = [x_i, x_{feat,i}, \beta^i]^T$, where $x_i \in \mathbb{R}^{3n}$ is the physical state of the robot moving in $\mathbb{R}^3$, typically made of position and velocity, i.e., $n = 2$, while $x_{feat,i}$ and $\beta^i \in \{0,1\}^{n_r}$ are the additional states used for the estimation of the global behaviour, with dynamics modelled as decoupled single integrators and influenced by the multi-agent behaviour. Regarding the consensus protocol, depending on the approach adopted, different considerations can be done on the transient time. Here we present the classical consensus protocol [13], given by:

$$u_h = \sum_{k\in\mathcal{N}_h} a_{hk}(z_k - z_h) \tag{4}$$

where z_h is a component of either $x_{feat,i}$ or β^i. A simple and reliable heuristics is to consider the consensus achieved after an amount of time larger than five times the time-constant τ associated to the Laplacian matrix $\mathcal{L}$, dictated by its second smallest eigenvalue λ_2. See also [12] for a similar application.

Remark 1 (Inflation): The final consensus state z_h^* is a function of the initial states, depending on the left zero eigenvector of the Laplacian matrix, whereas in principle both the mapping A and the state β are binary quantities. For the sake of consistency, a normalization procedure of the final state value is needed, named *inflation*:

$$z_h^* = \begin{cases} 1 & \text{if } z_h^* \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Furthermore, the connectivity of $\mathcal{G}$, resulting in $\lambda_2 > 0$, plays a fundamental role in the consensus performance.

Let us summarize the previous results in a formal statement addressing decentralized task allocation.

Theorem 1: Let us consider the multi-agent system $\{q_i\}_{i=1}^{n_r}$, communicating over an undirected and connected graph $\mathcal{G}$. The optimization problem (2)*, supported by Algorithm 1 and Algorithm 2 with consensus law given by* (4)*, enables the system to perform task allocation and fault detection in a fully decentralized fashion, thus providing a solution to Problem 1.* •

Proof: We first observe that decentralized fault detection is naturally addressed by Algorithm 2. For the case of task allocation, decentralized execution can be achieved as follows. First, Algorithm 1 runs until a full agreement is reached, requiring a transient time equal to 5τ. The full agreement condition $x_{feat,i} = x_{feat,j}$ $\forall i,j = 1,...,n_r$ allows each robot to execute a local version of the optimization scheme (2) using the same set of parameters, thus leading to coinciding allocation results among all the robots. ■

III. PROPOSED FRAMEWORK

In this section we present the overall scheme defining a decentralized task and control allocation for heterogeneous overactuated multi-robot systems, which exploits the benefits coming from both agent redundancy and actuator redundancy. Assume that the dynamics of each robot x_i is modelled as an overactuated double integrator of the form

$$\ddot{x}_i = -k\dot{x}_i + B_i u_i = -k\dot{x}_i + v_i \tag{6}$$

where $v_i \in \mathbb{R}^n$ is the virtual control input, k is a damping factor, $B_i \in \mathbb{R}^{n\times r_i}$ is the full row-rank input matrix and $u_i \in \mathbb{R}^{r_i}$ is the actual control input, with $r_i > n$.

A. Task execution

The robots are asked to perform both constant and time-varying regulation tasks, thus the corresponding CBF depends on both position and velocity of the robot i. Due to the dynamic structure of the model (6), we introduce a class of time-varying CBFs with cross terms in order to have a well-posed constraint (3b), i.e., ensuring $L_g h(x,t) \neq 0$ at time $t = 0s$, thus resulting in the following quadratic form

$$h_m(x,t) = -\frac{1}{2}\left\|[x_i - x_d(t) \;\; \dot{x}_i - \dot{x}_d(t)]\right\|_P^2 \tag{7}$$

where the symmetric matrix $P = P^T > 0$ is positive definite and non-diagonal. Accordingly, the constraint (3b) in the optimization problem can be exploited as follows:

$$\frac{\partial h_m(x,t)}{\partial x}\dot{x} + \frac{d}{dt}h_m(x,t) \geq -\gamma(h_m(x,t)) - \delta_{im} \tag{8}$$

where

$$L_g h_m(x,t) = p_{12}(x_i - x_d(t)) + p_{22}(\dot{x}_i - \dot{x}_d(t)) \tag{9}$$

with p_{12} and p_{22} being the components of the matrix P. It is worth noticing that, even though the velocity is initialized with a matched condition $\dot{x}_i(0) = \dot{x}_d(0)$, the term (9) is still different from zero since $x_i(0) \neq x_d(0)$ and the entry $p_{12} \neq 0$ because P is non-diagonal. In conclusion, the fulfillment of the constraint (8), within the resolution of the QP-problem (3), provides the virtual control v_i for the execution of the assigned task.

B. Adaptive control allocation

Bearing the static relationship $v_i = B_i u_i$ in mind, a control allocation problem can be formalized in terms of a quadratic programming problem of the form [14]:

$$\begin{aligned} \underset{u_i\in\mathbb{R}^{r_i}}{\text{minimize}} \quad & \frac{1}{2}||u_i||^2_{\Delta_i} \\ \text{subject to} \quad & v_i = B_i u_i \end{aligned} \tag{10}$$

where $\Delta_i \in \mathbb{R}^{r_i \times r_i}$ is a positive definite weighting matrix, representing a desired usage of each single component defining u_i. Assuming B_i to be full row-rank, then the explicit solution of (10) reads as $u_i = \mathcal{C} v_i$ where

$$\mathcal{C} = \Delta_i^{-1} B_i^T (B_i \Delta_i^{-1} B_i^T)^{-1} \tag{11}$$

is a generalized pseudo-inverse that can be derived from the optimality conditions of (10) using Lagrange multipliers. In real applications, each input is subject to saturation constraints. Referring to the actuator dynamics

$$\dot{u}_i = -\eta_i u_i + w_i \tag{12}$$

these constraints can be on the amplitude $|u_i|$ as well as on the rate $|w_i|$. Defining the admissible sets

$$\begin{aligned} \mathcal{U}_i &= \{u_i \in \mathbb{R}^{r_i} : u_{i,min} \le u_{ij} \le u_{i,max}, j = 1, \dots, r_i\} \\ \mathcal{W}_i &= \{w_i \in \mathbb{R}^{r_i} : w_{i,min} \le w_{ij} \le w_{i,max}, j = 1, \dots, r_i\} \end{aligned} \tag{13}$$

where $w_{i,min}$ and $w_{i,max}$ are the rate bounds, we look for control inputs satisfying $(u_i, w_i) \in \mathcal{U}_i \times \mathcal{W}_i$, which yields the extended optimization problem

$$\begin{aligned} &\underset{u_i \in \mathbb{R}^{r_1}}{\text{minimize}} \ \frac{1}{2} ||u_i||^2_{\Delta_i} \\ &\text{subject to} \ \ v_i = B_i u_i, \ \ (u_i, w_i) \in \mathcal{U}_i \times \mathcal{W}_i \end{aligned} \tag{14}$$

The inclusion of saturation constraints in optimization problems like (14) may lead to unfeasible solutions. Advanced tools, such as anti-windup schemes (see for instance [15]), generate an additional reference for the controller to prevent unstable behaviours when the limits are exceeded. Unfortunately, such schemes are not well suited for QP-based control design. The solution of (14) involves mapping the virtual control v_i to the actual control u_i, which must adapt to possible faults while satisfying the input limitations. To this end, [16] reformulate the control allocation problem into a *model reference adaptive control problem*, whose modularity decouples the closed loop performances from the control allocation ones. They parametrically define the mapping and constrain the values to evolve within a limited range using a projection algorithm. We have extended that method by including a desired control component usage, encoded by the matrix Δ_i, adopting the Γ-projection algorithm [17, Section 4]. This algorithm allows the parameters to evolve towards a desired value, encoded by a suitable matrix Γ. The unconstrained problem (10) provides the minimum error solution, expressed through the weighted pseudo-inverse $\mathcal{C}$ in (11) and, consistently, we have made the choice $\Gamma = \mathcal{C}$.

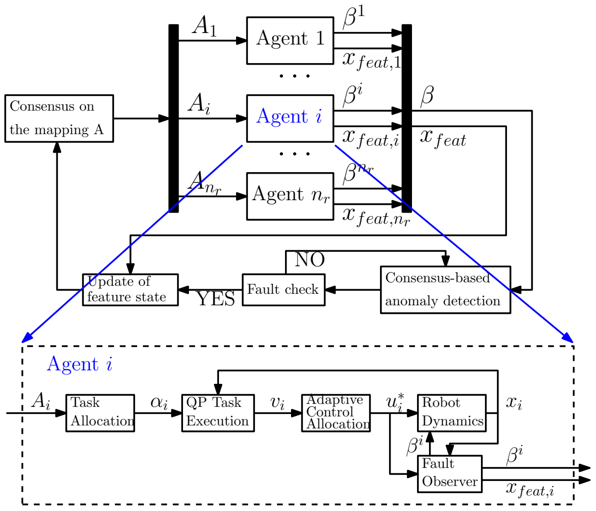


Fig. 2: Control scheme of the proposed framework.

C. Overall control scheme

We present here a third Algorithm 3, which describes the procedure that each agent has to follow in order to achieve the desired multi-robot system performance.

Algorithm 3 Resilient Task Allocation

```
Inputs:
Adjacency matrix $\mathcal{A}$
Number of robots $n_r$ and number of features $n_f$
Tasks $h_m, m \in \{1, \dots, n_t\}$
Mappings $H_k, T$
Parameters $n_{r,m,min}, n_{r,m,max}, \delta_{max}, c, l$
 1: for each robot $i$ do
 2:     Consensus on the mapping $A$                    ▷ Algorithm 1
 3:     Construction of the matrices $F$ and $S$
 4:     Evaluation of the matrix $\alpha$                ▷ (2)
 5:     while true do
 6:         Evaluation of the control $v_i$               ▷ (3)
 7:         Fault check                                  ▷ Algorithm 2
 8:         if there is a fault then
 9:             go to step 2
10:         end if
11:         Evaluation of the mapping for $v_i$ to $u_i$
12:         Execution of $u_i$ for completing the task
13:     end while
14: end for
```

From the above procedure, we can see that the occurrence of a fault, detected through the Algorithm 2, i.e., $\beta^i_j \neq 0$, triggers a new consensus cycle on the mapping A, since there has been a change in the robot capabilities. When a task re-allocation is called, robots need to stop and wait until the consensus in Algorithm 1 is achieved. To this end, we keep the value of $\beta^i_j = 1$ until the estimation of the mapping A has been terminated, then we reset the value of $\beta^i_j = 0$. Accordingly to the heuristics adopted, in a faulty situation we may end up in having a delay of 10τ between the detection of the fault and the execution of the control. The robot dynamics (6) can be enhanced and rewritten by including this operational condition

$$\ddot{x}_i = -k\dot{x}_i + (1 - \beta^i_i) B_i u_i \tag{15}$$

and so, whenever $\beta^i_i = 1$ no control is effectively executed as the robots are idle in a *situation awareness estimation* phase.

Theorem 2: Let consider a system Σ defined by a team of heterogeneous overactuated robots modelled as (6)*, with the control admissible sets*

$$\begin{aligned} \mathcal{U}_i &= \{u_i \in \mathbb{R}^{r_i} : u_{i,min} \le u_{ij} \le u_{i,max}, j = 1, \dots, r_i\} \\ \mathcal{W}_i &= \{w_i \in \mathbb{R}^{r_i} : w_{i,min} \le w_{ij} \le w_{i,max}, j = 1, \dots, r_i\} \end{aligned}$$

and a weighting matrix $\Delta_i \in \mathbb{R}^{r_i \times r_i}$. The cascade framework depicted in Figure 2, along with Algorithm 3, enables the robots to perform a resilient task allocation based only on local information, while simultaneously allocating the control according to the desired usage Δ_i, such that $(u_i(t), \dot{u}_i(t)) \in \mathcal{U}_i \times \mathcal{W}_i$. •

IV. SIMULATIONS

A. Problem setup

The performances of our framework have been tested on a team of seven robots, five of which are unicycles while the other two are UAVs. The system has to perform several tasks: t_1 is the coverage of a circular region, depicted by a red circle in the simulations, centred in the robot performing the second task t_2, tracking of a desired trajectory, and t_3 is the achievement of a desired hover position. The fulfillment of each task requires that the robots possess a certain set of features, hence of capabilities. The features available are wheels (f_1), a LIDAR with a limited proximity range d (f_2), a camera with a limited FOV (f_3) and a set of air propellers and a 360° camera (f_4). The capabilities required for performing the tasks are surveillance (c_1), capacity of following a line (c_2) and the ability to fly (c_3). Figure 1 reports the corresponding heterogeneity hypergraph.
We have tested the method in a faulty scenario, exploiting both the robot redundancy and the actuator redundancy. Faults can be of two types, either a robot may no longer be able to perform a task (e.g. loss of a feature) or one of the actuators of a robot may not work anymore, with the entries of a column of the matrix B, corresponding to the faulty input, being replaced by zeroes [7]. In our simulation results, we have considered the classic consensus protocol (4) since, in view of the *inflation* operation (5), we do not need the exact steady-state value to be reached by the multi-agent system, as we just care about the depletion of the transient.

B. Overactuation

Under a suitable dynamic feedback linearization, all the robots can be modelled as (6). In particular, concerning the UAVs, we have restricted the control variables to the Cartesian position (x, y, z) and we have considered each propeller as an independent actuator. The input matrices that we have considered are the following:

$$B_{un} = \begin{bmatrix} 1 & 0 & 0.5 \\ 0 & 1 & 0.5 \end{bmatrix} \quad B_{UAV} = \begin{bmatrix} -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{bmatrix} \tag{16}$$

The unicycles are characterized by three controls for the (x, y) dynamics, while the UAVs are endowed with four controls for the (x, y, z) dynamics. Regarding the control allocation block, the weighting matrices used for mapping properly the virtual control onto the actual control are respectively $W_{un} = \text{diag}([1, 5, 100])$ and $W_{UAV} = \text{diag}([1, 1, 1, 1])$. For the unicycles the extra component of the control, the third one, is penalized more in order to avoid undesired behaviours due to coupling effects. On the other hand, the weights for the UAVs have been chosen uniform, since the four propellers are all equivalent. Moreover, the unicycles have two different types of actuators: the first and second ones are just affected by saturation limits, while the third actuator has only rate limits. Since the control is the solution of a quadratic program, we have considered the following actuator dynamics for the third component of the control u: $\dot{u}_{i3} = -k_{prop}(u_{i3} - u^*_{i3})$, with $i = 1, 2, 3, 4, 6$, where u^*_{i3} is the third component of the output of the adaptive allocation block and $k_{prop} > 0$.

C. Task allocation parameters

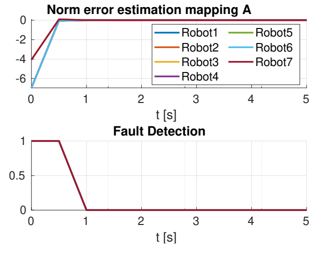
For the coverage task three robots are required, while only one robot is needed for the other two tasks. According to the features each agent possesses, robots r_2, r_3 and r_6 are assigned to task t_1, robot r_4 is assigned to task t_2, and the UAV_2 has to perform the hovering task t_3. The robot r_1 and the UAV_1 are left unemployed and so they remain idle at their initial positions. The regulation tasks t_j, with $j \in \{1, 2, 3\}$, are formalized with the following CBF

$$h_{i,j}(x, t) = -\frac{1}{2} \left\| [x_i - x_{dj}(t) \;\; \dot{x}_i - \dot{x}_{dj}(t)] \right\|^2_P$$

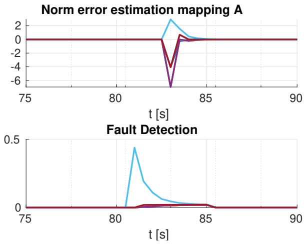
where $i \in \{1, \ldots, 7\}$. For task t_1, $x_{d1}(t) = G_i(x, t)$ and $\dot{x}_{1d}(t) = \dot{G}_i(x, t)$, with $G_i(x, t)$ being the centroid of the Voronoi cell that the ith robot has to cover, defined in [18]. Conversely, for task t_3, the desired position is constant $x_{d3}(t) = x_{hov}$ and, therefore $\dot{x}_{d3}(t) = 0$.

D. Simulation results

During the execution of the tasks two different faults occur. At time $t_1 = 80s$ the sixth unicycle breaks, loosing both the features it had at the beginning, hence a new assignment of the tasks is needed. Also robot r_1 has lost its feature f_2, as it is too far from robot r_4 to sense it. Then, the second UAV switches from performing task t_3 to executing the coverage with the other two unicycles, while the first UAV is assigned to the hovering task. Robot r_1 remains unemployed for the whole simulation.
In Figure 4 we display the control profile of all the robots of the team and the time evolution of the derivative of the third component of the unicycle control, which is affected by rate saturation. In Figure 3 we show the time evolution of the states β and x_{feat}, where the dashed grid identifies intervals of 5τ seconds. In both sub-figures, we can state that $\beta^i_i = 1$ until the consensus on the features has been concluded, then it is set to zero. In Figure 3b, after $t_1 = 80s$, the fault being locally diagnosed, the robot r_6 set the corresponding $\beta^6_6 = 1$, leading to a change in the values of the β state of the other agents. After 5τ seconds the fault occurrence has been detected globally, triggering a new consensus on the mapping A. On the other hand, at time $t_2 = 130s$, the first actuator of the third unicycle breaks, but the entity of the fault is mild enough to keep the robot able to perform its task, hence a new task allocation in not needed. In this case, the fault is directly managed by the adaptive control



(a) Initial consensus on the mapping A



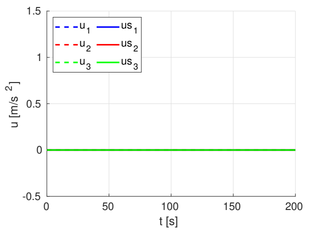
(b) New consensus on the mapping A after a fault

Fig. 3: Consensus protocol.

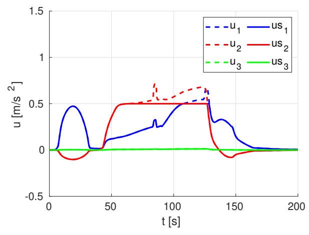
allocation block that reconfigures the mapping from v to u by avoiding the use of the faulty component. Figure 4c reports the control profile of the third unicycle, highlighting the change in the allocation policy. Furthermore, a video showing the simulation results in CoppeliaSim is available at [19].

V. CONCLUSIONS

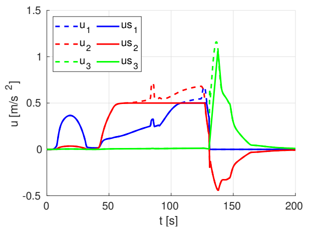
In this paper we have proposed a decentralized framework able to assign several tasks to a heterogeneous overactuated multi-robot system, while dealing with actuator limitations proper of each single agent. The framework results in a cascade of quadratic programs that each single agent can solve easily and independently of the others, ensuring scalability and decentralization. A consensus protocol has been included for both the acknowledgment of the feature mapping and for the detection of anomalies affecting in the system. We have tested the robustness of the method with respect to faults, modelled either as a complete change in the robot capabilities or as a malfunction of one of the actuators, showing that the team of robots successfully executed all the tasks assigned, adapting itself to the changes. Future works will be devoted to dealing with a more general architecture for nested and cumulative tasks, as well as assessing the robustness of the decentralized architecture to potential information loss during agent communications. Moreover, it is worth investigating the extension of the framework to more complex, possibly nonlinear, agent dynamics.



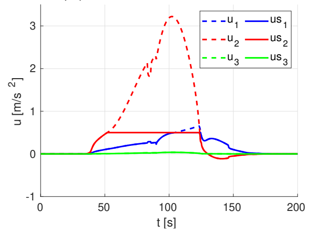
(a) Control Robot 1



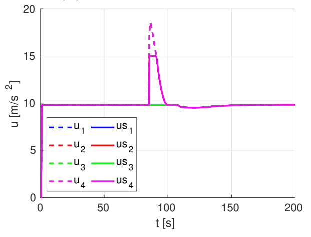
(b) Control Robot 2



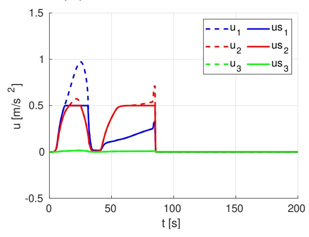
(c) Control Robot 3



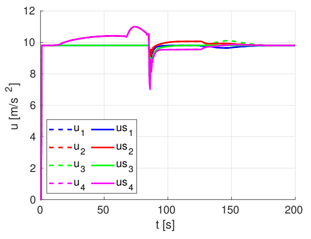
(d) Control Robot 4



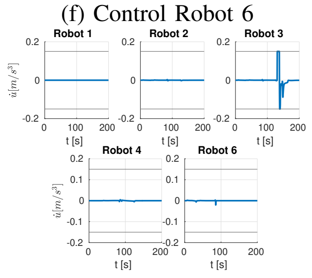
(e) Control UAV 1



(f) Control Robot 6



(g) Control UAV 2



(h) Derivative of u_{i3}

Fig. 4: Control profile: at $t_1 = 80s$ the 6^{th} unicycle breaks and at $t_2 = 130s$ the first actuator of the 3^{rd} unicycle breaks.

REFERENCES

[1] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 939–954, 2004.

[2] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," *Coop. robots and sensor networks*, pp. 31–51, 2015.

[3] G. Notomista, S. Mayya, Y. Emam, C. Kroninger, A. Bohannon, S. Hutchinson, and M. Egerstedt, "A resilient and energy-aware task allocation framework for heterogeneous multirobot systems," *IEEE Trans. Robot.*, vol. 38, no. 1, pp. 159–179, 2021.

[4] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 912–926, 2009.

[5] W. E. Walsh and M. P. Wellman, "A market protocol for decentralized task allocation," in *Proc. Int. Conf. on Multi Agent Systems*. IEEE, 1998, pp. 325–332.

[6] M. Egerstedt, J. N. Pauli, G. Notomista, and S. Hutchinson, "Robot ecology: Constraint-based control design for long duration autonomy," *Annual Reviews in Control*, vol. 46, pp. 1–7, 2018.

[7] A. Cristofaro and T. A. Johansen, "Fault tolerant control allocation using unknown input observers," *Automatica*, vol. 50, no. 7, pp. 1891–1897, 2014.

[8] ——, "Fault-tolerant control allocation with actuator dynamics: finite-time control reconfiguration," in *53rd IEEE Conf. Decis. Control*. IEEE, 2014, pp. 4971–4976.

[9] L. Govoni and A. Cristofaro, "A fault-tolerant task allocation framework for overactuated multi-robot systems," in *IEEE 9th Int. Conf. Control Decis. Inf. Technol.*, 2023, pp. 287–292.

[10] G. Notomista and M. Egerstedt, "Constraint-driven coordinated control of multi-robot systems," in *Proc. Amer. Control Conf.* IEEE, 2019, pp. 1990–1996.

[11] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th Eur. Control Conf.* IEEE, 2019, pp. 3420–3431.

[12] A. Cristofaro, G. Giunta, and P. R. Giordano, "Fault-tolerant formation control of passive multi-agent systems using energy tanks," *IEEE Syst. Control Lett.*, vol. 6, pp. 2551–2556, 2022.

[13] R. O. Saber and R. M. Murray, "Consensus protocols for networks of dynamic agents," in *Proc. Amer. Control Conf.* IEEE, 2003, pp. 951–956.

[14] T. A. Johansen and T. I. Fossen, "Control allocation—a survey," *Automatica*, vol. 49, no. 5, pp. 1087–1103, 2013.

[15] T. A. Lima, S. Tarbouriech, F. G. Nogueira, and B. C. Torrico, "Co-design of dynamic allocation functions and anti-windup," *IEEE Syst. Control Lett.*, vol. 5, no. 6, pp. 2198–2203, 2020.

[16] S. S. Tohidi, Y. Yildiz, and I. Kolmanovsky, "Adaptive control allocation for over-actuated systems with actuator saturation," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5492–5497, 2017.

[17] E. Lavretsky and T. E. Gibson, "Projection operator in adaptive systems," *arXiv ePrint arXiv:1112.4232*, 2011.

[18] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 243–255, 2004.

[19] L. Govoni and A. Cristofaro. Decentralized task allocation for redundant multi-robot systems: an iterative consensus approach. [Online]. Available: https://youtu.be/9v35xHyCyQk