Managing Conflicting Tasks in Heterogeneous Multi-Robot Systems Through Hierarchical Optimization

Davide De Benedittis, *Student Member, IEEE*, Manolo Garabini, *Member, IEEE*, and Lucia Pallottino, *Senior Member, IEEE*

***Abstract*—The robotics research community has explored several model-based techniques for multi-robot and multi-task control. Through constrained optimization, robot-specific characteristics can be taken into account when controlling robots and accomplishing tasks. However, in scenarios with multiple conflicting tasks, existing methods struggle to enforce strict prioritization among them, allowing less important tasks to interfere with more important ones. In this letter, we propose a novel control framework that enables robots to execute multiple prioritized tasks concurrently while maintaining a strict task priority order. The framework exploits hierarchical optimization within a model predictive control structure. It formulates a convex minimization problem in which all the tasks are encoded as linear equality and inequality constraints. The proposed approach is validated through simulations using a team of heterogeneous robots performing multiple tasks.**

***Index Terms*—Constrained motion planning, motion and path planning, multi-robot systems, path planning for multiple mobile robots or agents.**

I. Introduction

THE transition of robotic systems from more structured environments, such as laboratories and factories, to less structured ones is already in progress. Robots are now being deployed in more complex and dynamic environments, such as natural habitats [1], public spaces [2], and even hospitals [3]. These settings introduce unpredictable environmental conditions, unknown dynamics, and complex task requirements. The need for adaptable, robust, and efficient control strategies becomes crucial, particularly in multi-robot systems tasked with concurrently executing multiple objectives that conflict with each other [4]. In such scenarios, robots must not only navigate unpredictable environments but also coordinate their actions to achieve overarching goals.

In the case of multi-robot fleets deployed for long-term duties, the ability to dynamically adapt their behavior in response to environmental, task, and robot changes is essential [5]. Additionally, heterogeneous multi-robot systems, where different robots have distinct capabilities and specialties, offer promising solutions for a wide range of applications, such as search and rescue [6], environmental monitoring [7], and logistics [8]. Finally, the use of these systems in delicate environments, like hospitals, requires solutions that can guarantee safety and reliability through hardware and software. Therefore, the development of strategies that can manage the complexity of heterogeneous fleets executing multiple concurrent tasks, while ensuring safety and efficiency, is of paramount importance.

In this letter, we propose a model-based control framework designed for heterogeneous multi-robot systems tasked with executing multiple concurrent objectives. Our approach leverages Quadratic Programming (QP) and the Hierarchical Optimization (HO) framework to formulate a Model Predictive Controller (MPC) that ensures optimal task execution that adapts to changes and manages system-level constraints. The main contributions of this work are: 1. a novel MPC framework capable of executing multiple concurrent tasks while strictly enforcing task priority hierarchies (Section III), 2. the formalization of several multi-robot tasks within the proposed framework (Section IV). Additional contributions include: 3. a comparison with the state-of-the-art, demonstrating the advantages of the proposed approach (Section V-A), 4. the validation on a team of heterogeneous robots performing conflicting tasks in diverse scenarios (Section V-B), and 5. a quantitative analysis of the computational cost and scalability of the proposed approach (Section VI).

A. Related Works

Multiple solutions have been proposed to address the control of multi-robot systems performing multiple tasks. Below, we review the most relevant methods that deal with task prioritization and conflict resolution, explaining their limitations in relation to the proposed framework.

Artificial Potential Fields (APF)-based methods offer a computationally efficient means of multi-robot coordination. APF rely on a weighted summation of potential fields, leading to inter-task interference and violation of physical constraints. For instance, [9] applies APF to formation control and obstacle

Received 2 December 2024; accepted 1 April 2025. Date of publication 10 April 2025; date of current version 21 April 2025. This article was recommended for publication by Associate Editor L. Liu and Editor M. A. Hsieh upon evaluation of the reviewers' comments. This work was supported in part by the European Union Next Generation EU project under Grant ECS00000017 "Ecosistema dell'Innovazione" Tuscany Health Ecosystem (THE, PNRR, Spoke 9: Robotics and Automation for Health), in part by the MUR Program PRIN 2022, project HOME4.0 under Grant E53D23000510006, and in part by the Italian Ministry of University and Research (MUR) in the framework of the FoReLab and CrossLab projects (Department of Excellence). *(Corresponding author: Davide De Benedittis.)*

The authors are with Centro di Ricerca "E. Piaggio", Università di Pisa, 56122 Pisa, Italyand also with the Dipartimento di Ingegneria dell'Informazione, Universitá di Pisa, 56122 Pisa, Italy (e-mail: davide.debenedittis@phd.unipi.it).

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2025.3559843, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3559843

avoidance; however, due to the absence of strict prioritization, the success rate heavily depends on parameter tuning. Moreover, APF-based approaches struggle to integrate system constraints and are prone to local minima.

Optimization-based frameworks provide a more structured approach. In [10], [11], the Authors use Control Barrier Functions (CBFs) to guarantee constraint satisfaction and ensure long-term autonomy. While CBFs offer strong theoretical properties, they handle prioritization through soft constraints within a weighted optimization problem. As a result, task execution depends on appropriate parameter tuning, and strict prioritization is not guaranteed.

Similarly, in [12], the Authors propose an approach for solving task assignment in multi-robot systems. It ensures that all tasks are completed efficiently using local communication but, due to the presence of a task assigner, it does not consider the presence of concurrent tasks at the planning level.

Graph-based strategies inspired by information-invariant theory [13] relax the single-task assumption and allow robots to handle synergistic tasks concurrently. However, these approaches focus primarily on task allocation and scheduling rather than motion planning.

Null-Space-based Optimization (NSO) [14] enforces prioritization by recursively projecting tasks into the null space of higher-priority objectives. However, [14] employs classical NS methods capable of handling only equality constraints, limiting its suitability for a broader range of tasks. Without a structured prioritization mechanism—like the proposed HO framework—NSO lacks the flexibility required to handle complex multi-task motion planning scenarios.

Unlike weighted methods like [9], [11], we introduce Hierarchical Optimization (HO) [15], [16] to multi-robot, multi-task motion planning, explicitly enforcing strict task prioritization while ensuring feasibility. Differently from [14], we handle both equality and inequality constraints and optimize over longer horizons using MPC. Additionally, while [14] only compares classical NSO methods against each other, we compare our approach with the state-of-the-art weighted approach—weighted optimization—evaluating its advantages in terms of strict prioritization and computational efficiency.

II. Problem Formulation

A. Robot Models

We consider a fleet of n_{r} heterogeneous robots, whose nonlinear dynamics are described by

$$\dot{s}_i = f_i(s_i, u_i), \tag{1}$$

where $s_i \in \mathbb{R}^{n_{s_i}}$ and $u_i \in \mathbb{R}^{n_{u_i}}$ are the state and input vectors of the ith robot, $\dot{s}_i \in \mathbb{R}^{n_{s_i}}$ is the time derivative of the state vector, and f_i is a locally Lipschitz continuous vector field. We assume that each robot has access to the relative positions of other robots and obstacles, either by a centralized unit or, equivalently, by local sensors.

The robots' dynamics are discretized and linearized around the state and input at the kth instant ($\bar{s}_{i,k}$ and $\bar{u}_{i,k}$), i.e.:

$$\tilde{s}_{i,k+1} = A_{\text{dyn},i,k}\tilde{s}_{i,k} + B_{\text{dyn},i,k}\tilde{u}_{i,k}, \tag{2}$$

where $\tilde{s}_{i,k} = s_{i,k} - \bar{s}_{i,k}$, $\tilde{u}_{i,k} = u_{i,k} - \bar{u}_{i,k}$, while $A_{\text{dyn},i,k} = \frac{\partial f_i}{\partial s_i}(\bar{s}_{i,k}, \bar{u}_{i,k})$, and $B_{\text{dyn},i,k} = \frac{\partial f_i}{\partial u_i}(\bar{s}_{i,k}, \bar{u}_{i,k})$ are the discrete time dynamics matrices.

B. Task Model

The multi-robot team is required to execute a set of m tasks $\bar{\mathcal{T}} = \{\mathcal{T}_1, \ldots, \mathcal{T}_m\}$. The tasks will guide the robotic fleet's behavior by constraining the collection of the states and inputs of all the robots, i.e., $x = \begin{bmatrix} s_{1,k} & \ldots & s_{n_{\mathrm{r}},k} & u_{1,k} & \ldots & u_{n_{\mathrm{r}},k} \end{bmatrix}$.

The pth task is formulated as a feasibility problem of a set of equality and inequality constraints, i.e.,

$$\mathcal{T}_p : \begin{cases} A_p x - b_p = 0, \\ C_p x - d_p \leq \mu_p, \end{cases} \tag{3}$$

where, $p = 1, \ldots, m$ while the matrices A_p, b_p, C_p, and d_p are appropriately sized matrices and vectors w.r.t the optimization variable x. The slack variable μ_p, together with the relaxation of the equality constraint—minimized in the cost function in Section III-A—ensure the feasibility of the problem.

III. Solution Approach

In this section, we describe the solution approach for addressing the problem of multi-robot control with multiple concurrent tasks $\bar{\mathcal{T}}$. Hierarchical Optimization (HO) is introduced in Section III-A, which ensures that the tasks are executed according to an importance order, guaranteeing that higher-priority ones are not compromised by lower-priority tasks. Afterward, Section III-B describes the MPC problem that, leveraging HO, computes the optimal states and inputs of the fleet over a finite time horizon.

A. Hierarchical Optimization

Consider a set of tasks $\bar{\mathcal{T}} = \{\mathcal{T}_1, \ldots, \mathcal{T}_m\}$ where each task takes the form in (3). These tasks are ordered according to a user-defined priority level. For compactness, in this section we assume that $\mathcal{T}_p$ is of higher importance than $\mathcal{T}_{p+1}$; however, in general, the priority level can be independent from the task index. The core idea behind hierarchical optimization is the iterative computation of x_p^*, by solving the task $\mathcal{T}_p$ and all the higher-priority inequality constraints. Since equality constraints of higher priorities are not satisfied by x_p^*, it is referred to as a *partial* solution.

Conversely, we refer to $\bar{x}_p^*$ as the *global* solution that verifies the equality (and inequality) constraints of all higher-priority tasks. The *global* solution is computed recursively by updating the previous *global* solution with the projection of the partial solution x_p^* in the null space of the higher-priority equality constraints (as in (4)). It is worth noting that this solution is guaranteed to satisfy, within physical constraints, all the tasks optimally while also preventing less important tasks from affecting more important ones.

More formally, $\bar{x}_p^*$ is recursively computed with

$$\bar{x}_p^* = \bar{x}_{p-1}^* + N_{p-1}x_p^*, \tag{4}$$

initialized with $\bar{x}_0^* = 0$.

The nullspace projector N_p is calculated with an efficient recursive method as follows

$$N_p = N_{p-1}\,\mathrm{Null}(A_p N_{p-1}), \tag{5}$$

initialized with $N_0 = I$ and where, given a matrix M and its singular value decomposition $M = U\Sigma V^{\mathbf{T}}$, with Σ having singular values in decreasing order, the null space is

$$\mathrm{Null}(M) = V_{:,r+1:n_x}, \tag{6}$$

where $V_{:,r+1:n_x}$ are the last $n_x - r$ columns of V and r is the number of singular values above a given threshold τ, i.e.,

$$r = \sum_{i=1}^{n_x} 1(\sigma_i > \tau), \tag{7}$$

Note that the number of columns of N_p decreases as tasks constrain the solution space.

The *partial* solution x_p^* is obtained by solving the following QP problem

$$\begin{bmatrix} x_p^{*\mathbf{T}} & \mu_p^{*\mathbf{T}} \end{bmatrix} = \underset{\xi_p = [x^{\mathbf{T}} \ \mu_p^{\mathbf{T}}]^{\mathbf{T}}}{\arg\min} \quad \frac{1}{2}\xi_p^{\mathbf{T}} Q_p \xi_p + p_p^{\mathbf{T}} \xi_p$$
$$\text{s.t.} \qquad \widehat{\mathrm{C}}_{\mathrm{p}} \xi_{\mathrm{p}} \leq \widehat{\mathrm{d}}_{\mathrm{p}}, \tag{8}$$

where

$$Q_p = \begin{bmatrix} N_{p-1}^{\mathbf{T}} A_p^{\mathbf{T}} A_p N_{p-1} + \sigma I & 0 \\ 0 & I \end{bmatrix}, \tag{9}$$

$$p_p = \begin{bmatrix} N_{p-1}^{\mathbf{T}} A_p^{\mathbf{T}} (A_p^{\mathbf{T}} \bar{x}_{p-1}^* - b_p) \\ 0 \end{bmatrix}, \tag{10}$$

$$\widehat{C}_p = \begin{bmatrix} C_p N_{p-1} & -I \\ \bar{C}_{p-1} N_{p-1} & 0 \end{bmatrix}, \tag{11}$$

$$\widehat{d}_p = \begin{bmatrix} d_p - C_p \bar{x}_{p-1}^* \\ \bar{d}_{p-1} - \bar{C}_{p-1} \bar{x}_{p-1}^* + \bar{\mu}_{p-1}^* \end{bmatrix}. \tag{12}$$

For more details on the derivation of the QP problem, we refer the reader to [15], [17]. The positive scalar σ in (9) is a regularization factor that guarantees that Q_p is positive definite. It does not derive directly from the hierarchical formulation, but improves convergence properties and is required by some numerical QP solvers. Its value is chosen to avoid numerical issues and to prevent it from affecting the solution. Additionally, the matrices $\bar{C}_p$, $\bar{d}_p$, and $\bar{\mu}_p^*$ are the vertical stack of the corresponding matrices or vectors from the highest priority (i.e., priority 1) to p, i.e.,

$$\bar{C}_p = \begin{bmatrix} C_1 \\ \vdots \\ C_p \end{bmatrix}, \quad \bar{d}_p = \begin{bmatrix} d_1 \\ \vdots \\ d_p \end{bmatrix}, \quad \bar{\mu}_p^* = \begin{bmatrix} \mu_1^* \\ \vdots \\ \mu_p^* \end{bmatrix}. \tag{13}$$

Concluding, the solution of the whole set of tasks $\bar{\mathcal{T}}$ is $\bar{x}_m^*$, obtained after solving m different QP subproblems and projecting their solutions into the null space of the higher-priority tasks, as in (4).

B. Hierarchical Constraint-Based Model Predictive Control

We tackle the problem of multi-robot constraint-based control by employing HO in an MPC framework. The MPC covers a time horizon of $n_{\mathrm{c}}\Delta t$, where n_{c} is the number of control steps and Δt is a constant sampling time. Both n_{c} and Δt consider the trade-off between temporal resolution, prediction horizon, and computational complexity.

The optimization problem employs the Direct Multiple Shooting (DMS) method, which jointly optimizes the states and the inputs of the robots. Despite the increase in the number of optimization variables compared to Direct Single Shooting (DSS) (which optimizes only over the inputs), DMS provides better convergence and does not suffer from numerical instability [18].

We define the collection of the variational discrete-time states and inputs of the multi-robot system along the optimization horizon as

$$\begin{aligned} \tilde{s} &= \begin{bmatrix} \tilde{s}_{1,1}^{\mathbf{T}} & \dots & \tilde{s}_{1,n_{\mathrm{c}}}^{\mathbf{T}} & \dots & \tilde{s}_{n_{\mathrm{r}},1}^{\mathbf{T}} & \dots & \tilde{s}_{n_{\mathrm{r}},n_{\mathrm{c}}}^{\mathbf{T}} \end{bmatrix}^{\mathbf{T}}, \\ \tilde{u} &= \begin{bmatrix} \tilde{u}_{1,0}^{\mathbf{T}} & \dots & \tilde{u}_{1,n_{\mathrm{c}}-1}^{\mathbf{T}} & \dots & \tilde{u}_{n_{\mathrm{r}},0}^{\mathbf{T}} & \dots & \tilde{u}_{n_{\mathrm{r}},n_{\mathrm{c}}-1}^{\mathbf{T}} \end{bmatrix}^{\mathbf{T}}, \end{aligned} \tag{14}$$

where $\tilde{s}_{i,k}$ and $\tilde{u}_{i,k}$ are the variations of the state and input vector of the ith robot at the kth instant with respect to the linearization point. Note that linearizing around a nominal trajectory instead of a point significantly improves the algorithm's performance and reduces linearization errors.

The full optimization vector of the MPC problem is

$$x = \begin{bmatrix} \tilde{s}^{\mathbf{T}} & \tilde{u}^{\mathbf{T}} \end{bmatrix}^{\mathbf{T}}. \tag{15}$$

At each priority level p, the HO framework augments x with the slack variable μ_p associated with $\mathcal{T}_p$ (as in (8)).

Having defined the optimization vector, the HO-based MPC problem is fully defined by the set of tasks $\bar{\mathcal{T}}$, along with their priority levels. This set of tasks must not only shape the robot's behavior but also ensure the solution's consistency with the robots' dynamics. Therefore, the robots' dynamics are formulated as a linear task and assigned the highest priority. Additionally, the input limits task has a high priority to ensure that the commands respect the actuators' limits. The remaining tasks are ordered according to the specific application requirements. This implicitly modular design allows for easy modification of the tasks and their priorities, making the proposed approach flexible and adaptable to different scenarios. Furthermore, no parameter tuning is required, but only priority ordering.

IV. Tasks Formulation

The proposed approach builds upon the concept of tasks, which are encoded as equality and inequality constraints in the optimization problem. This formulation is suitable for representing a broad class of multi-robot tasks relevant to real-world applications. This section presents some example tasks for multi-mobile-robot systems and their formulation as in (3). These tasks will be later used in the validation section to compare the proposed approach with the state-of-the-art and to demonstrate its capabilities. It is important to note that the tasks in (3) do not need both equality and inequality constraints, as the matrices corresponding to the missing constraints can be empty matrices. Additionally, the proposed approach can be seen as a superset of weighted optimization, since multiple tasks can be assigned the same priority level and solved simultaneously through relative weighting.

A. Robot Dynamics and Inputs Limit Task

The proposed approach handles heterogeneous multi-robot systems, where heterogeneity can be in terms of task execution capabilities, dynamics, or constraints. The robots' dynamics are encoded as a task by linearizing them around a reference state and input. The discretization is performed using a first-order forward Euler. After linearization, the dynamics constraint task is expressed as in (2).

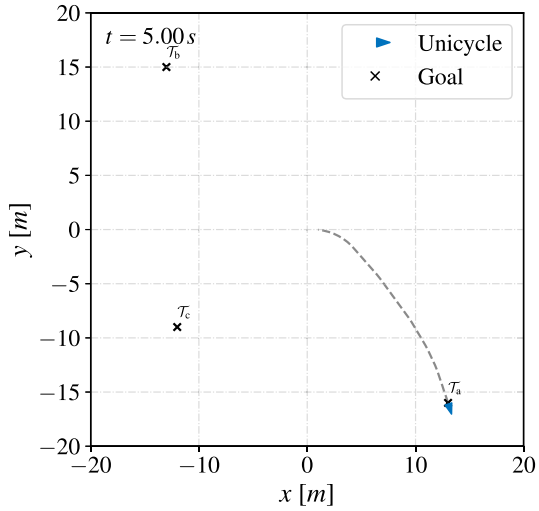


Fig. 1. Conflicting go-to-goal tasks scenario solved with the proposed approach. $\mathcal{T}_a$ is the task with the highest priority, followed by $\mathcal{T}_b$ and finally $\mathcal{T}_c$.

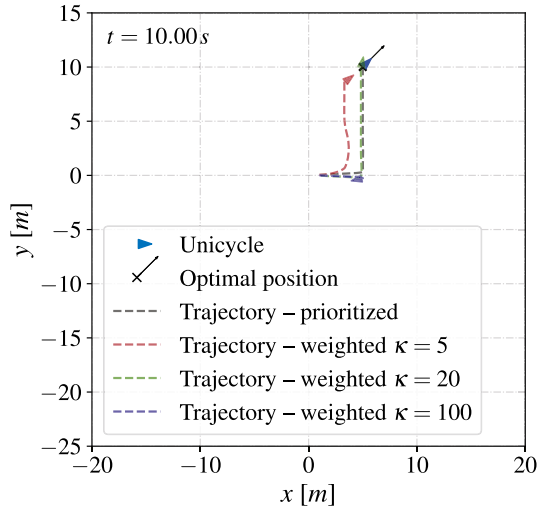


Fig. 2. Path executed by the robot with the two different control methods: prioritized and weighted approach.

TABLE I
KPIs OF THE CONFLICTING GO-TO-GOAL TASKS SCENARIO

Method	Weights [κ]	Time-to-Goal	Position Error
Prioritized	/	**4.05 s**	**0.00 m**
Weighted	10	4.09 s	0.62 m
Weighted	100	4.05 s	0.01 m
Weighted	1000	4.05 s	0.00 m

For instance, considering a unicycle model with states $s = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$ and inputs $u = \begin{bmatrix} v & \omega \end{bmatrix}^T$, the discretized dynamics task is formulated as

$$\begin{aligned} x_{k+1} &= x_k + v_k \cos(\theta_k)\Delta t, \\ y_{k+1} &= y_k + v_k \sin(\theta_k)\Delta t, \\ \theta_{k+1} &= \theta_k + \omega_k \Delta t. \end{aligned} \tag{16}$$

Here, Δt is the sampling time, while the subscript k denotes the time step.

Additionally, the limits on the input vector are encoded as inequality constraints as follows

$$u_{i,\min} \leq u_{i,k} \leq u_{i,\max}, \tag{17}$$

$$\Delta u_{i,\min} \leq u_{i,k} - u_{i,k-1} \leq \Delta u_{i,\max}, \tag{18}$$

where (17) represents the limits on the input vector, while (18) enforces the limits on the input rate.

B. Collision Avoidance Task

Collision avoidance, a crucial task in mobile robot systems, is formulated by enforcing a minimum safety distance d_s between robots and obstacles (or other robots).

When the obstacle position does not depend on the optimization vector, the task is encoded as a linear inequality constraint as

$$(r_{i,k} - r_{\text{obs},k})^{\mathbf{T}}(\bar{r}_{i,k} - r_{\text{obs},k}) \geq d_s^2, \tag{19}$$

where $r_{i,k}$, $\bar{r}_{i,k}$, and $r_{\text{obs},k}$ are the positions of the ith robot, its linearization at time step k, and the obstacle position at time step k, respectively.

Conversely, (linearized) collision avoidance between two generic robots i and j is formulated as

$$((r_{i,k} - r_{j,k})^{\mathbf{T}}(\bar{r}_{i,k} - \bar{r}_{j,k}) \geq d_s^2. \tag{20}$$

C. Go-to-Goal Task

The go-to-goal task requires the robot to reach a specific position in the environment. It is formulated as an equality task as follows

$$r_{i,k} = r_{\text{goal}}, \tag{21}$$

where r_{goal} is the goal position.

D. Coverage Task

Coverage is a common task in multi-robot systems where robots are tasked with spreading over a domain in an optimal way. As shown in [19], optimality is achieved by considering Voronoi cells and minimizing

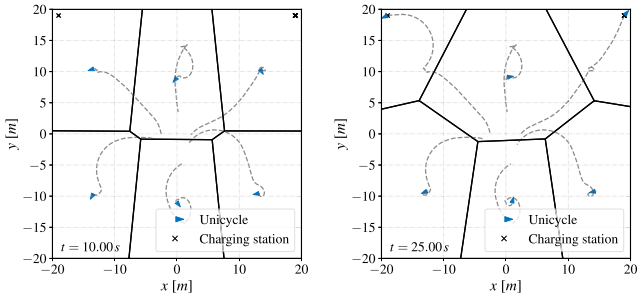
$$J_{\text{cov}} = \sum_{i=1}^{N} \int_{q \in \mathcal{V}_i} \|q - r_{i,k}\|^2 \phi(q) dq. \tag{22}$$

Here, $\phi(\cdot) : \mathcal{Q} \to \mathbb{R}$ is a density function that maps each point $q \in \mathcal{Q}$ of the environment to a scalar value, while $\mathcal{V}_i$ is the Voronoi cell associated to robot i. Due to coverage being an NP-hard problem and its globally optimal solution difficult to obtain, a locally optimal solution is often used. Indeed, by taking the gradient of J_{cov}, a local optimal solution for minimizing the cost can be found as

$$r_i^* = \frac{\int_{q \in \mathcal{V}_i} q\phi(q) dq}{\int_{q \in \mathcal{V}_i} \phi(q) dq} = c_{\mathcal{V}_i}, \tag{23}$$

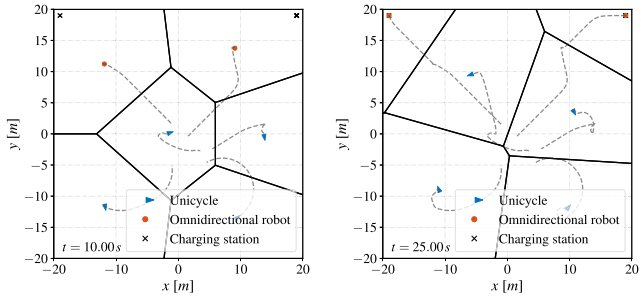
where $c_{\mathcal{V}_i}$ is the centroid of the Voronoi cell $\mathcal{V}_i$. It translates into a task that can be written as a linear equality task (also known as the Lloyd formula) as follows

$$r_{i,k} = c_{\mathcal{V}_{i,k}}. \tag{24}$$



(a) Coverage only. (b) Coverage and go-to-goal.

Fig. 3. Coverage task (from the start) and battery charge tasks (from 15 s onwards) with six unicycle robots.



(a) Coverage only. (b) Coverage and go-to-goal.

Fig. 4. Coverage task (from the start) and battery charge tasks (from 15 s onwards) with a heterogeneous fleet.

E. Formation Control Task

Formation control is yet another frequent task in multi-robot systems in which the robots are required to maintain a specific geometric configuration. The configuration is specified by means of a set of relative positions between the robots. It can be encoded as a linear equality task as follows

$$\frac{(r_{i,k} - r_{j,k})^{\mathbf{T}}(\bar{r}_{i,k} - \bar{r}_{j,k})}{\|\bar{r}_{i,k} - \bar{r}_{j,k}\|} = d_{ij}, \qquad \forall i, j \in \mathcal{N}, \tag{25}$$

where d_{ij} is the desired distance between robots i and j and $\mathcal{N}$ defines all the pairs of robots for which a desired distance is given. It is worth noting that the formation must be admissible in order to be achieved. However, since the proposed approach uses slack variables, it is capable of handling infeasible formations by relaxing the constraints, avoiding the optimization problem becoming infeasible.

F. Input Minimization Task

The input minimization task is useful for reducing the robots' energy consumption. It can be encoded in the cost function, by minimizing the inputs' 2-norm, or as an equality constraint, by setting the input vector to zero. While the first approach would compromise all the tasks' execution, encoding it as the lowest priority task avoids jeopardizing all the other tasks. Therefore, the constraint is as follows:

$$u_{i,k} = 0. \tag{26}$$

V. Validation

In this section, we compare the proposed approach with the state of the art (Section V-A) and validate it through simulations in multiple scenarios (Section V-B).

A. Comparison With the State of the Art

Here, the proposed prioritized approach is compared with the weighted one within the framework of optimization-based control strategies. The weighted approach solves the same tasks simultaneously, using a weighted sum of the tasks' costs, similarly to [11], [17]. Two simplified scenarios, when considered together, highlight the differences between the methods and the advantages of the prioritized one.

1) Conflicting Go-to-Goal Tasks: We first consider a scenario involving a single unicycle robot required to perform a set of three *conflicting* and persistent go-to-goal tasks, denoted as $\mathcal{T}_\mathrm{a}$, $\mathcal{T}_\mathrm{b}$, and $\mathcal{T}_\mathrm{c}$. Each task consists in maintaining a different position, with $\mathcal{T}_\mathrm{a}$ having a higher importance than the others. This specification is encoded in the hierarchical formulation by setting the priority of the task $\mathcal{T}_\mathrm{a}$ as 1 and the priorities of $\mathcal{T}_b$ and $\mathcal{T}_\mathrm{c}$ as 2. In contrast, the weighted framework handles task importance through relative task weight s, using a κ parameter to be properly tuned. Since a weight of zero would completely exclude a task from consideration, we compare the robot's behavior obtained with the proposed HO-based approach with the weighted approach using three different values of the parameter κ: 10, 100, and 1000.

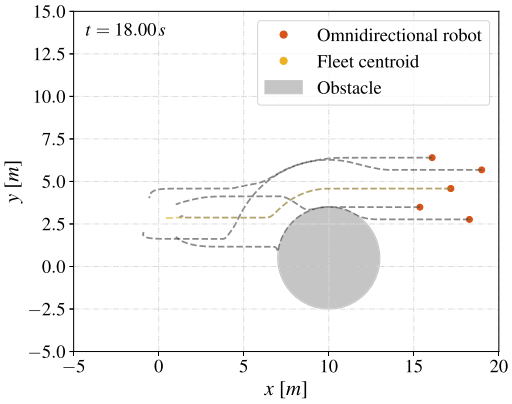
Fig. 1 represents the three goals and the robot's trajectory when controlled by the proposed approach. It shows that the robot moves smoothly and in accordance with its dynamics while satisfying $\mathcal{T}_\mathrm{a}$ without compromise. Additionally, Table I reports the time to reach the goal and the final position error with the different approaches. The prioritized approach outperforms the weighted approaches with $\kappa = 10$ and $\kappa = 100$, achieving lower time-to-goal and lower position error. At $\kappa = 1000$, the results of the weighted control law match those of the HO-based approach.

2) Multiple Prioritized Tasks: We now consider again a single robot scenario in which a unicycle robot is required to perform a more complex task set $\mathcal{T}$ that comprises *partially conflicting* tasks. The tasks do constrain only some coordinates of the robot states: $\mathcal{T}_1 : x = 5, \mathcal{T}_2 : x = -12, \mathcal{T}_3 : y = 10$, $\mathcal{T}_4 : y = -4$, and $\mathcal{T}_5 : \theta = \pi/4$. The tasks have a decreasing importance, with $\mathcal{T}_1$ being the most important and $\mathcal{T}_5$ the least. It follows that the optimal final configuration is $x = 5$, $y = 10$, and $\theta = \pi/4$.

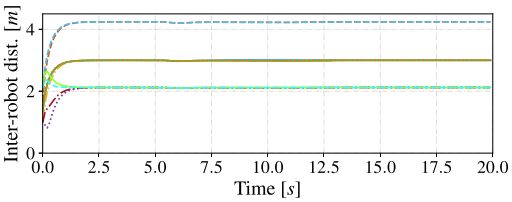
Fig. 2 shows the trajectory followed by the robot using the HO-based approach (in black) and with the weighted approach ($\kappa = 5$, 20, 100 in red, green, and blue, respectively).

3) Discussion: The results in Table I emphasize the crucial role of the κ parameter in the weighted approach. Even in simple scenarios, some level of tuning is required, and the method exhibits high sensitivity to parameter values.

Although the weighted method performs similarly to the HO-based approach when $\kappa = 1000$, the "Multiple Prioritized Tasks" scenario disproves the assumption that increasing κ can yield results comparable to the prioritized approach. In fact, as shown in Fig. 2, with high values of κ (i.e., 100) the robot completes only one task, even when others are non-conflicting. Moreover, low values of κ (e.g., 5) lead to the robot failing to accomplish any tasks, and even "optimal" values (e.g., 20) result in compromises and uncompleted feasible tasks ($\mathcal{T}_5$). Notably, no weight value perfectly replicates the desired behavior achieved with HO.

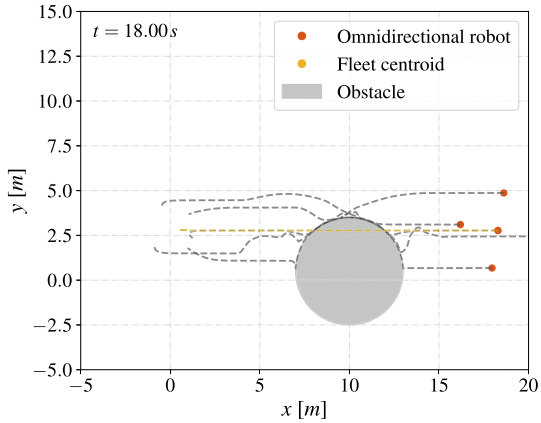


(a) Robots' trajectories and scenario overview.

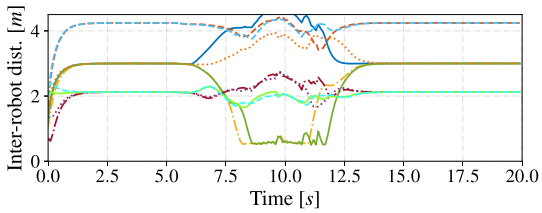


(b) Inter-robot distances over time.

Fig. 5. Fleet of five omnidirectional robots tasked with (from highest to lowest priority) collision avoidance, formation control, and centroid velocity reference.



(a) Robots' trajectories and scenario overview.



(b) Inter-robot distances over time.

Fig. 6. Fleet of five omnidirectional robots tasked with (from highest to lowest priority) collision avoidance, centroid velocity reference, and formation control.

In contrast, the proposed approach achieves the desired behavior without parameter tuning. The robot first moves horizontally to satisfy $\mathcal{T}_1$, ensuring minimal delay in its completion by not addressing other tasks simultaneously. Only after $\mathcal{T}_1$ is satisfied does it move vertically to complete $\mathcal{T}_3$, followed by a final rotation to accomplish $\mathcal{T}_5$.

B. Validation in Multi-Robot Multi-Task Scenarios

To further validate the proposed approach, we consider more complex scenarios involving fleets of heterogeneous mobile robots. Two distinct scenarios are analyzed: a coverage task combined with battery charging requests, and a situation requiring both formation control and centroid velocity tracking, while also performing collision avoidance.

1) Coverage and Battery Charging Tasks: We consider a heterogeneous robot fleet composed of six robots, including both unicycles and omnidirectional robots. The fleet must perform coverage of a square area and each robot must maintain sufficient battery charge (which is recharged by reaching a charging station). The coverage task is prioritized when the robot's battery is full while, when it reaches a certain threshold, recharging is assigned a higher priority. This allows the remaining robots to continue the coverage task, adapting to the movement of the recharging robot. The charging task is modeled as a go-to-goal task (see (21)).

We evaluate the results of this scenario with two robotic fleets: one composed of six unicycle robots (Fig. 3) and the other of four unicycles and two omnidirectional robots (Fig. 4). Fig. 3(a) shows the unicycle positions and past trajectories at $t = 10\,\mathrm{s}$, when they have reached a good coverage of the area and the batteries have not reached the charging threshold. Fig. 3(b), acquired at $t = 25\,\mathrm{s}$, shows how the charging stations have been reached by the two robots with low batteries while the coverage is still guaranteed by the other four robots. The same experiment is repeated with the heterogeneous fleet composed of four unicycles and two omnidirectional robots. The evolutions are reported in Figs. 4(a) and 4(b), which demonstrate that the approach is robust to robot heterogeneity and can seamlessly manage different robot types by considering their dynamics and constraints in the optimization problem.

2) Collision Avoidance, Formation Control, and Centroid Velocity Reference Tasks: We consider a robot fleet composed of five omnidirectional robots. The fleet is tasked with maintaining a specific formation and following a centroid velocity reference. Additionally, an obstacle is present in the environment. Therefore, the fleet is also tasked with obstacle and self-collisions avoidance. Two task priority configurations are tested out.

a) Formation before centroid reference: In this scenario, the collision avoidance has the highest priority, followed by the formation control (requiring robots on a square's vertexes and one in the center), and, finally, the centroid velocity reference $v_{\mathrm{r}} = \begin{bmatrix} 1 & 0 \end{bmatrix}^{\mathbf{T}}$, see Fig. 5. The self-collisions avoidance is encoded by maintaining a distance between robots of at least $50\,\mathrm{cm}$. Fig. 5(a) shows the robots' trajectories (in black), the centroid's trajectory in yellow, and the scenario overview. Fig. 5(b) displays the inter-robot distances over time.

b) Centroid reference before formation: In the second scenario, represented in Fig. 6, the centroid velocity reference has a higher priority than the formation task, while collision avoidance has the highest priority. As in previous scenario, Fig. 6(a) shows the robots' trajectories, while Fig. 6(b) shows the inter-robot distances over time.

c) Discussion: The results demonstrate that the prioritized approach effectively handles conflicting tasks, ensuring that

TABLE II
SOLUTION TIMES OF THE APPROACHES

Scenario	Setup Time	Solve Time	
		Hierarchical	Weighted
Conflicting go-to-goal	0.106 s	0.121 s	0.031 s
Multiple prioritized tasks	0.102 s	0.174 s	0.056 s
Coverage	1.6 s	4.19 s	2.72 s
Collision + Centroid + Formation	0.45 s	0.74 s	1.05 s

the most critical (and safety-related) ones are achieved without compromise.

When formation control is prioritized over the centroid velocity reference (Fig. 5), the obstacle is successfully avoided while maintaining the formation, even during evasive maneuvers. As a consequence, the velocity reference task is momentarily violated when necessary but followed whenever possible. In this case, the maximum 2-norm centroid velocity error reaches 0.7 m/s.

Conversely, when the centroid velocity reference is prioritized over formation control (Fig. 6), the robots accurately track the reference velocity even while avoiding the obstacle. Here, the 2-norm of the velocity error remains 0 m/s at all times. However, the formation is temporarily broken when required to satisfy higher-priority tasks.

A key advantage of the prioritized approach is its robustness and parameter-free nature. Unlike weighted optimization based methods, which require careful tuning of task weights, our method enforces prioritization via null-space projections, inherently preserving safety and task hierarchy. This ensures reliable performance across different environmental conditions. In contrast, classical weighted approaches rely on manual weight tuning, which directly affects safety. For instance, [9] investigates a similar scenario where task success rates—particularly for collision avoidance—vary significantly based on task difficulty and parameter tuning.

VI. COMPUTATION TIME ANALYSIS

Here, we present and compare the computation times required to control the robot fleet in the scenarios presented in Section V. Afterward, we describe how the number of robots, the prediction horizon, and the constraints affect the computational cost. The results are obtained using a computer with a 12th Gen Intel Core i7-12700H CPU. The algorithm is implemented in Python 3.10, while tasks are defined and linearized using the CasADi library [20]. Several QP solvers were tested and the best results, from a computational point of view, were obtained using the Python `quadprog` solver.

The computation times of the prioritized and weighted approaches are reported in Table II. The times are averaged over three runs of the centralized algorithms, using $n_c = 4$ time steps in the MPC. The values refer to the whole time required to solve the tasks for its whole duration (15 s or 25 s, depending on the scenario), not a single optimization problem. The computation times are divided into two parts: 1. the "setup", which includes the time to create the matrices and define the optimization problem, and 2. the "solve", which is the time to solve the optimization problem. Note that the setup time is approximately the same for both approaches.

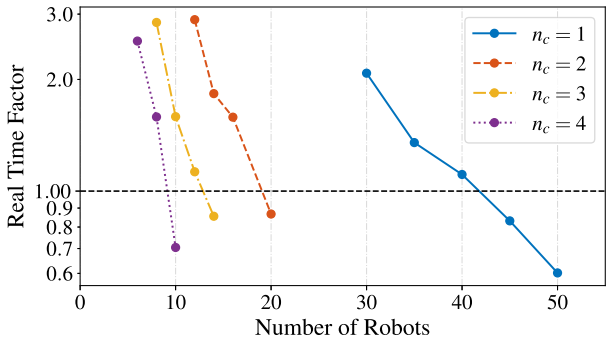


Fig. 7. Real-time factor when varying n_r and n_c.

From a computational point of view, the main difference between the weighted and the hierarchical approach is that they solve a different number of problems of different sizes. The weighted approach solves a single optimization problem of size $\text{len}(x) + \sum_{p=1}^{m} \text{rows}(C_p)$, where $\text{len}(x)$ is the dimension of the optimization variable and $\text{rows}(C_p)$ is the number of rows of the matrix C_p. Conversely, the hierarchical approach solves multiple subproblems, each corresponding to a different priority level, of (at worst) size $\text{len}(x) + \text{rows}(C_p)$. Note that, due to (6), lower priority tasks have a smaller-sized $x \in \mathbb{R}^{n_x - r}$, since N_p gets progressively fewer columns. Additionally, the optimization times also depend on the constraints (both equality and inequality) and their number, which can heavily affect the convergence speed.

The weighted approach is faster than the prioritized one in the first three cases, where inequality constraints are limited in number. More specifically, the weighted approach is approximately four times faster than the hierarchical one in the first two cases, and 50% faster in the third case. On the other hand, the hierarchical approach is faster in the last case, where it outperforms the weighted approach by 40%.

The worse performance of the hierarchical approach in the first three cases derives from the fact that the tasks have a "small" number of inequality constraints (compared to the dimension of the optimization vector x). The performance difference between the first two scenarios and the third one (where the HO approach performs better) is due to the different number of constraints of the problems solved by the weighted and hierarchical approach. Conversely, the fourth case has a higher number of inequality constraints (due to the collision avoidance task), and is solved by the hierarchical approach through multiple subproblems of smaller size than the weighted one. This leads to the hierarchical approach being faster than the weighted one in this case.

We also analyze the algorithm performance when varying the number of robots and the prediction horizon n_c. We consider the scenario in Section V-B1 applied to omnidirectional robots; Fig. 7 shows the real-time factor (RTF) (the ratio of the time to formulate and solve the QP and the period of the MPC) of the hierarchical approach with different n_r and n_c, when the MPC frequency is 20 Hz. It can be observed that, depending on the prediction horizon, the algorithm can be real-time capable with fleets of more than 40 robots.

Concluding, the performance of the HO-based approach is heavily dependent on the number and distribution of both equality and inequality constraints. In general, in the worst case, the hierarchical approach can be up to m times slower. On the other hand, with a favorable constraints distribution, the hierarchical approach outperforms the weighted one from a computational point of view, in addition to providing more guarantees. Note that the computational complexity scales linearly with n_c and quadratically with n_{r} and with the dimension of concatenation of the input and state vector s (i.e., $\sum_{i+1}^{n}(n_{s_i} + n_{u_i})$).

VII. Conclusion

This letter presented a novel hierarchical constraint-based model predictive control framework for heterogeneous multi-robot systems. By leveraging hierarchical optimization within a model predictive control structure, the proposed method efficiently handles task prioritization, ensuring strict task ordering and execution while simultaneously considering robot dynamics and constraints. Through simulations, the framework was validated in a variety of multi-task, multi-robot scenarios, demonstrating its superior performance compared to state-of-the-art weighted optimization approaches.

Key findings include the framework's ability to maintain task priority without requiring extensive parameter tuning, unlike traditional weighted approaches. Furthermore, it is suitable for heterogeneous robot systems and complex environments, efficiently managing multiple concurrent tasks such as collision avoidance, coverage, formation control, and battery charging. Overall, the proposed hierarchical optimization approach offers a flexible and computationally efficient solution for multi-robot systems, enabling robust control in dynamic and conflicting task scenarios.

Future work will explore decentralized optimization and the validation under more dynamic conditions.

References

[1] F. Angelini et al., "Robotic monitoring of habitats: The natural intelligence approach," *IEEE Access*, vol. 11, pp. 72575–72591, 2023.

[2] A. H. While, S. Marvin, and M. Kovacic, "Urban robotic experimentation: San Francisco, Tokyo and Dubai," *Urban Stud.*, vol. 58, no. 4, pp. 769–786, 2021.

[3] S. Jeon, J. Lee, and J. Kim, "Multi-robot task allocation for real-time hospital logistics," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2017, pp. 2465–2470.

[4] Y. Jiang, H. Yedidsion, S. Zhang, G. Sharon, and P. Stone, "Multi-robot planning with conflicts and synergies," *Auton. Robots*, vol. 43, no. 8, pp. 2011–2032, 2019.

[5] B. Li, B. R. Page, B. Moridian, and N. Mahmoudian, "Collaborative mission planning for long-term operation considering energy limitations," *IEEE Robot. Automat. Lett.*, vol. 5, no. 3, pp. 4751–4758, Jul. 2020.

[6] D. Yanguas-Rojas and E. Mojica-Nava, "Exploration with heterogeneous robots networks for search and rescue," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 7935–7940, 2017.

[7] I. Lončar et al., "A heterogeneous robotic swarm for long-term monitoring of marine environments," *Appl. Sci.*, vol. 9, no. 7, 2019, Art. no. 1388.

[8] Y. Rizk, M. Awad, and E. W. Tunstel, "Cooperative heterogeneous multi-robot systems: A survey," *ACM Comput. Surv.*, vol. 52, no. 2, pp. 1–31, 2019.

[9] W. Liu, H. Zhang, J. Hu, M. Y. Wang, and Z. Xiong, "A distributed formation controller with multi-obstacle avoidance for multi-mobile robot system," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatron.*, 2022, pp. 256–261.

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

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

[12] X. Bai, W. Yan, M. Cao, and D. Xue, "Distributed multi-vehicle task assignment in a time-invariant drift field with obstacles," *IET Control Theory Appl.*, vol. 13, no. 17, pp. 2886–2893, 2019.

[13] W. Smith and Y. Zhang, "Achieving multitasking robots in multi-robot tasks," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 8948–8954.

[14] A. Mannucci, D. Caporale, and L. Pallottino, "On null space-based inverse kinematics techniques for fleet management: Toward time-varying task activation," *IEEE Trans. Robot.*, vol. 37, no. 1, pp. 257–274, Feb. 2021.

[15] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 785–792, Aug. 2011.

[16] D. De Benedittis, F. Angelini, and M. Garabini, "Soft bilinear inverted pendulum: A model to enable locomotion with soft contacts," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 55, no. 2, pp. 1478–1491, Feb. 2025.

[17] S. Xin, "Hierarchical QP," Accessed: Feb. 20, 2025. [Online]. Available: https://sites.google.com/site/xinsongyan/research/hierarchical-qp

[18] P. Puchaud, F. Bailly, and M. Begon, "Direct multiple shooting and direct collocation perform similarly in biomechanical predictive simulations," *Comput. Methods Appl. Mechan. Eng.*, vol. 414, 2023, Art. no. 116162.

[19] W. Luo and K. Sycara, "Voronoi-based coverage control with connectivity maintenance for robotic sensor networks," in *Proc. IEEE Int. Symp. Multi-Robot Multi-Agent Syst.*, 2019, pp. 148–154.

[20] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi–A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, 2019.