Generalized Assignment for Multi-Robot Systems via Distributed Branch-And-Price

Andrea Testa, Giuseppe Notarstefano

Abstract

In this paper, we consider a network of agents that has to self-assign a set of tasks while respecting resource constraints. One possible formulation is the Generalized Assignment Problem, where the goal is to find a maximum payoff while satisfying capability constraints. We propose a purely distributed branch-and-price algorithm to solve this problem in a cooperative fashion. Inspired by classical (centralized) branch-and-price schemes, in the proposed algorithm each agent locally solves small linear programs, generates columns by solving simple knapsack problems, and communicates to its neighbors a fixed number of basic columns. Also, we propose a cloud-assisted version of the algorithm that accelerates the branching process and requires less memory and computation from agents. We prove finite-time convergence of both the algorithms to an optimal solution of the problem. Then, we apply the proposed schemes to a dynamic task assignment scenario in which a team of robots has to visit tasks appearing over time. We implement the proposed algorithms in a ROS testbed and provide experiments for a team of heterogeneous robots solving the dynamic assignment problem.

I. Introduction

The *Generalized Assignment Problem* (GAP) is a well known combinatorial optimization problem with several applications as vehicle routing, facility location, resource scheduling and supply chain, to name a few [1], [2]. Even though GAP is a NP-hard problem, several approaches have been developed for solving this problem both for exact and approximate solutions. We refer the reader to [3] for a survey. Branch-and-price algorithms [2], [4] are among the most investigated algorithms allowing for both optimal and suboptimal solutions.

Task assignment naturally arises in cooperative robotics, where heterogeneous agents collaborate to fulfill a complex task, see, e.g., [5] for an early reference. Specific applications include persistent monitoring of locations [6], path planning of mobile robots, e.g., UAVs [7], task scheduling for robots working in the same space [8], vehicle routing [9] and task assignment in urban environments [10]. All the previous problems are solved by means of centralized approaches.

In order to deal with the computational complexity of the problem, a branch of literature analyzes parallel and decentralized approaches to the problem[1]. A well known parallel approach is the auction based one, originally proposed in [11]. A market-based approach is considered in [12] for the coordination of human-robot teams. Authors in [13] propose an algorithm, based on a sequential shortest augmenting path scheme, to solve a dynamic multi-task allocation problem. Agents propose assignments that are validated by a coordinating unit. As for decentralized schemes, authors in [14] solve a dynamic task allocation problem for robots that can perform local sensing operations and do not communicate among each other. In [15], a task assignment problem is solved, in a decentralized scheme, through the so called petal algorithm. In [16], a dynamic task assignment problem in which the cost vector changes in a bounded region is considered. A central unit is initially required but robots are able to exploit local communications to perform a reallocation if needed. An area partitioning problem for multi-robot systems

This result is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 638992 - OPT4SMART).

A. Testa and G. Notarstefano are with the Department of Electrical, Electronic and Information Engineering, University of Bologna, Bologna, Italy, a.testa@unibo.it, giuseppe.notarstefano@unibo.it.

[1]We denote *parallel* the approaches based on master-slave architectures, while we call *decentralized* the schemes with independent agents that do not communicate among each others.

is proposed in [17] and solved by a genetic algorithm. An area coverage problem in marine environments is solved in [18] with heuristics based on the traveling salesman problem.

The exact resolution of GAP in a purely distributed way, i.e., in a peer-to-peer network where processors communicate and perform local computations without a central coordinator, is an open problem. State-of-the-art solutions are usually based, as we discuss in the next, on proper linear programming relaxations or suboptimal approaches. In the context of multi-robot applications, a distributed version of the Hungarian method is proposed in [19]. A distributed simplex scheme for degenerate linear programs (LP) is proposed in [20] in the context of multi-agent assignment problems. In [21], authors apply a distributed dual subgradient method to the task assignment problem. Authors in [22] propose a distributed column generation algorithm for general LPs and time-varying communication graphs that can be applied to linear programming relaxations of task assignment problems. In these approaches, authors neglect integrality constraints on the decision variables, relying on the unimodular structure of the problems. As for distributed, suboptimal approaches for task assignment problems, in [23] a large-scale distributed task/target assignment problem across a fleet of autonomous UAVs is considered. A distributed column generation scheme under the assumption of complete communication graph is proposed, endowed with a round up procedure to retrieve a suboptimal integer solution. In the context of wireless sensor networks, [24] proposes a distributed task allocation in order to maximize the network life-time. A distributed task assignment algorithm is used in conjunction with a deterministic annealing in [25], in the context of limited-range sensor coverage. In [26], a dynamic vehicle routing problem is approached with a distributed protocol in which agents iteratively solve graph partitioning problems and communicate with neighboring agents. Distributed implementations of the auction-based algorithm are often used to solve task assigment problems, see, e.g., [27] for an early reference, and in particular GAPs [28], [29]. The auction-based approach allows for a suboptimal solution with performance guarantees. In [30] this approach is applied to a task allocation problem expressed as a combinatorial optimization problem with matroid constraints. In the recent works [31], [32], a dynamic task allocation scenario, allowing agents for a partial replanning is considered.

The contributions of this paper are as follows. We propose a purely distributed version of the branch-and-price algorithm to solve the Generalized Assignment Problem by means of a network of agents. Specifically, each agent locally solves a linear programming relaxation of the GAP, generates columns by solving a (simple) knapsack problem, and exchanges estimates of the solution with neighboring agents. Due to the relaxation of the integrality constraints, the solution of this problem may not be feasible. Thus, new problems, based on the original one with suitable additional constraints, have to be solved. The set of these problems can be represented by a so called branching tree. By leveraging on their communication capabilities, agents explore their local trees until an optimal solution of the optimization problem has been found. To the best of the authors' knowledge, this is the first attempt to solve GAP to optimality in a purely distributed fashion. As a variant, we provide a cloud-assisted version of the algorithm, which is useful in scenarios, as, e.g., cooperative robotics in smart factories, where an additional infrastructure with computation capability and storage may be available. In the proposed distributed algorithm, an aiding computing unit supports agents in the tree exploration, thus reducing their computation burden and memory requirements, and speeding up the algorithm convergence. Finally, we apply the proposed algorithm to a dynamic task assignment problem where tasks arrive during time. An experimental platform, based on ROS (Robot Operating System), is proposed to run experiments in which a team of aerial and ground robots cooperatively solve the dynamic task assignment problem relying on the proposed distributed branch-and-price scheme.

The paper unfolds as follows. In Section II we introduce the distributed setup considered throughout the paper. Then, we introduce the Generalized Assignment Problem and a centralized scheme, called *branch-and-price*, to solve it. In Section III we propose a purely distributed branch-and-price algorithm and a cloud-assisted version. In Section IV we provide numerical simulations for randomly generated GAPs and in Section V we show the results of experiments on a swarm of heterogeneous robots solving a dynamic task assignment problem.

Notation: We denote by e_ℓ the ℓ-th vector of the canonical basis (e.g., $e_1 = [1\,0\,\dots\,0]^\top$) of proper dimension. Given a vector $v_\ell \in \mathbb{R}^d$, we denote by v_{ℓ_m} the m-th component of v_ℓ. Also, we denote by 1_r (0_r) the vector in $\mathbb{R}^r$ with all its entries equal to 1 (0).

II. Distributed Setup and Preliminaries

In this section, we introduce the distributed setup for the Generalized Assignment Problem addressed in the paper. Also, the (centralized) branch-and-price scheme is illustrated.

A. Distributed Problem Setup

In the Generalized Assignment Problem, the objective is to find a maximal profit assignment of M tasks to N agents such that each task is assigned only to one agent. In this scenario, the generic agent i has a reward $p_{im} \in \mathbb{R}$ if it executes the m-th task. It also has a limited capacity $g_i \in \mathbb{R}$ and it uses an amount $w_{im} \in \mathbb{R}$ of capacity if it performs the m-th task. Let x_{im} be a binary variable indicating whether task m is assigned to agent i ($x_{im} = 1$) or not ($x_{im} = 0$). Then, the standard integer programming formulation is the following

$$\begin{aligned} \max_{x_{11},\dots,x_{NM}} \quad & \sum_{i=1}^{N}\sum_{m=1}^{M} p_{im}x_{im} \\ \text{subj. to} \quad & \sum_{i=1}^{N} x_{im} = 1, m = 1,\dots,M, \\ & \sum_{m=1}^{M} w_{im}x_{im} \le g_i, \forall i = 1,\dots,N, \\ & x_{im} \in \{0,1\}, i = 1,\dots,N, m = 1,\dots,M. \end{aligned} \tag{1}$$

In order to streamline the notation, we now introduce a formulation of the GAP better highlighting the structure of the problem in a distributed scenario. Let $z_i = [x_{i1},\dots,x_{iM}]^\top \in \mathbb{R}^M, \forall i = 1,\dots,N$. In the following we denote as z the stack $[z_1^\top,\dots,z_M^\top]^\top$. Also, let $c_i = [p_{i1},\dots,p_{iM}]^\top \in \mathbb{R}^M$, $D_i = [w_{i1},\dots,w_{iM}] \in \mathbb{R}^M$ and $P_i = \{z_i \in \{0,1\}^M \mid D_i z_i \le g_i\}$, for $i = 1,\dots,N$. Then, (1) can be recast as

$$\begin{aligned} \max_{z_1,\dots,z_N} \quad & \sum_{i=1}^{N} c_i^\top z_i \\ \text{subj. to} \quad & \sum_{i=1}^{N} z_i = 1_M, \\ & z_i \in P_i, i = 1,\dots,N. \end{aligned} \tag{2}$$

This new formulation of (1) allows us to point out the distributed nature of the problem. Namely, c_i describes the profits associated to assigning tasks to agent i, $\sum_{i=1}^{N} z_i = 1_M$ describes the assignment constraints (*coupling constraints*), P_i describes the capacity restrictions on the agents (*local constraints*). It is worth noting that the sets P_i are bounded for $i = 1,\dots,N$.

The agents must solve (2) cooperatively in a distributed fashion with limited communication and computation capabilities, as well as limited memory. We consider the natural scenario in which the i-th agent only knows the polyhedron P_i and the cost vector c_i, thus not having knowledge of other agent data. In order to solve the problem, agents can exchange information according to a time-varying communication network modeled as a time-varying digraph $\mathcal{G}^t = (\{1,\dots,N\}, \mathcal{E}^t)$, with $t \in \mathbb{N}$ being a universal slotted time representing a temporal information on the graph evolution. Notice that time t does not need to be known by the agents. A digraph $\mathcal{G}^t$ models the communication in the sense that there is

an edge $(i,j) \in \mathcal{E}^t$ if and only if agent i is able to send information to agent j at time t. For each node i, the set of *in-neighbors* of i at time t is denoted by $\mathcal{N}_{i,t}^{\text{in}}$ and is the set of j such that there exists an edge $(j,i) \in \mathcal{E}^t$. A static digraph is said to be *strongly connected* if there exists a directed path for each pair of agents i and j. Next, we require the following.

Assumption 2.1 (Graph Connectivity): The communication graph is L-strongly connected, i.e., there exists an integer $L \geq 1$ such that, for all $t \in \mathbb{N}$, the graph $(\{1,\dots,N\}, \bigcup_{\tau=t}^{t+L-1} \mathcal{E}^\tau)$ is strongly connected. □

Notice that this is a standard, mild, assumption in the context of distributed optimization that allows to model direct, time-varying, asynchronous and possibly unreliable communication.

B. Centralized Branch-and-Price Method

We now introduce the main concepts regarding the branch-and-price scheme. For the sake of clarity, we organize this subsection in three parts.

Dantzig-Wolfe Decomposition for GAPs

An equivalent formulation of (2), which is exploited in our distributed setup, can be obtained as follows. Such procedure, originally introduced in [4], is strictly related to the *Dantzig-Wolfe Decomposition* [33]. Points $z_i \in P_i$ can be represented as the linear combination of a finite number of vectors v_i^q, with $q \in \{1,\dots,|Q_i|\}$, i.e.,

$$z_i = \sum_{q=1}^{|Q_i|} v_i^q \lambda_i^q, \tag{3}$$

where we denote with Q_i the set of vectors v_i^q. The variables λ_i^q, $\forall q = 1,\dots,|Q_i|$, also called *combiners*, have to satisfy

$$\begin{aligned} &\sum_{q=1}^{|Q_i|} \lambda_i^q = 1, \\ &\lambda_i^q \in \{0,1\}. \end{aligned} \tag{4}$$

It can be shown that, for GAPs, Q_i coincides with the set of extreme points of the convex hull $\text{conv}(P_i)$ of P_i, [4]. Let $\Lambda \in \mathbb{R}^{\sum_{i=1}^N |Q_i|}$, be the stack of all the combiners. Substituting (3) and (4) in (2) leads to the following equivalent Integer Programming Master Problem (IP-MP)

$$\begin{aligned} \max_{\Lambda} \ & \sum_{i=1}^{N} \sum_{q=1}^{|Q_i|} (c_i^\top v_i^q) \lambda_i^q \\ \text{subj. to } & \sum_{i=1}^{N} \sum_{q=1}^{|Q_i|} v_i^q \lambda_i^q = 1_M, \\ & \sum_{q=1}^{|Q_i|} \lambda_i^q = 1, i = 1,\dots,N, \\ & \lambda_i^q \in \{0,1\}, q \in \{1,\dots,|Q_i|\}, i = 1,\dots,N. \end{aligned} \tag{5}$$

Notice that an optimal solution $z^\star$ of (2) can be retrieved from an optimal solution $\Lambda^\star$ of (5) by substituting the entries of $\Lambda^\star$ in (3), [2].

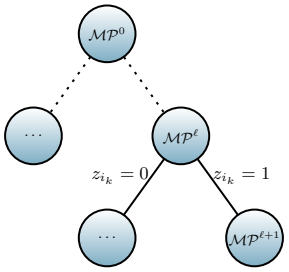


Fig. 1. Example of branching tree with problems generated according to constraints on the sets P_i.

Branching Tree

The presence of binary constraints makes (5) hard to solve. In order to find an optimal solution to the problem, the branch-and-price algorithm, [4], explores the set of feasible solutions of GAP by iteratively generating and solving *relaxed* versions of (5) including suitable, tightening constraints. That is, the constraint $\lambda_i^q \in \{0,1\}$ of all these problems is relaxed to $\lambda_i^q \geq 0$ ($\lambda_i^q \leq 1$ can be omitted as it is implicit in the constraint $\sum_{q=1}^{|Q_i|} \lambda_i^q = 1$). These problems can be represented as nodes of a so called *branching tree*, see, e.g., Figure 1. The ℓ-th node of the tree to be solved represents a problem, in the form of (5), obtained by relaxing the integer constraints and enforcing the constraints of the edges. We denote such problem as $\mathcal{MP}^\ell$, by $\Lambda^{\star\ell}$ its optimal solution and by $z^{\star\ell}$ and $J^{\star\ell}$ the solution and cost in terms of the variables z. Branches (edges) indicate the constraints that that have to been added to generate new problems. Instead of considering constraints on the Λ variables, new problems are generated by including, to the sets P_i, additional constraints in the form $z_{i_k} = 0, z_{i_k} = 1$ for some $z_{i_k}(\ell) \notin \{0,1\}$. In the following, we assume there exists an extraction strategy to determine the next node of the tree to be solved and a rule to choose the index i_k. Let P_i^ℓ be the sets obtained by including such additional constraints and let Q_i^ℓ be the sets of extreme points of $\text{conv}(P_i^\ell)$. Since $Q_i^\ell \subset Q_i$, this results in generating relaxed problems in the form of (5) with less optimization variables. The algorithm keeps track of a *lowerbound*, also called *incumbent*, J^{INC} on the cost, and of a candidate solution z^{INC}. After solving the generic problem $\mathcal{MP}^\ell$, one of the following operations is performed:

- *Incumbent Update:* If $z^{\star\ell} \in \{0,1\}^{NM}$ and $J^{\star\ell} \geq J^{\text{INC}}$, then $J^{\text{INC}} = J^{\star\ell}$ and $z^{\text{INC}} = z^{\star\ell}$.
- *Branching:* If $J^{\star\ell} > J^{\text{INC}}$ and $z^{\star\ell} \notin \{0,1\}^{NM}$ and $J^{\star\ell} \geq J^{\text{INC}}$, two new problems are added to the tree.
- *Pruning:* If $J^{\star\ell} \leq J^{\text{INC}}$ or $\mathcal{MP}^\ell$ is infeasible, nothing is done.

Remark 2.2: Pruning prevents the algorithm from inspecting problems that do not improve J^{INC}. □

At the end of the algorithm J^{INC} and z^{INC} coincide with the optimal cost $J^\star$ and solution $z^\star$ of (2).

Column Generation

Each problem $\mathcal{MP}^\ell$ has a large number of optimization variables. Thus, it can be approached by means of the so called *column generation algorithm* (originally proposed in [34] in the context of cutting stock problems). It consists in iteratively performing the following three steps:

i) A *Restricted Master Problem* (RMP) is solved, made by a small subset of the *columns*[2] of $\mathcal{MP}^\ell$

[2]We refer the reader to Appendix A for the definition of column for a linear program. Informally, a column is a portion of the cost and constraint vectors associated to a decision variable.

with sets $\bar{Q}_i^\ell \subset Q_i^\ell$ and a smaller combiner vector $\tilde{\Lambda}$, i.e.,

$$\begin{aligned} \max_{\tilde{\Lambda}} \ & \sum_{i=1}^{N} \sum_{q=1}^{|\bar{Q}_i^\ell|} (c_i^\top v_i^q) \tilde{\lambda}_i^q \\ \text{subj. to } & \sum_{i=1}^{N} \sum_{q=1}^{|\bar{Q}_i|} v_i^q \tilde{\lambda}_i^q = 1_M, \\ & \sum_{q=1}^{|\bar{Q}_i^\ell|} \tilde{\lambda}_i^q = 1, i = 1, \dots, N, \\ & \tilde{\lambda}_i^q \geq 0, q \in \{1, \dots, |\bar{Q}_i^\ell|\}, i = 1, \dots, N. \end{aligned} \tag{6}$$

ii) If possible, new columns are added to the RMP in order to improve the current solution. Let $[\pi^\top \ \mu^\top]^\top$ be a dual optimal solution of (6) at a generic iteration of the algorithm. In particular, $\pi \in \mathbb{R}^M$ is associated to the constraint $\sum_{i=1}^N \sum_{q \in \bar{Q}_i^\ell} v_i^q \tilde{\lambda}_i^q = 1_M$, while $\mu \in \mathbb{R}^N$ is associated to the constraint $\sum_{q \in \bar{Q}_i^\ell} \tilde{\lambda}_i^q = 1$. For each $i = 1, \dots, N$ a new column is found by solving the so called *pricing problem*:

$$\begin{aligned} \bar{v}_i \in \underset{z_i}{\text{argmax}} \ & (c_i - \pi)^\top z_i \\ \text{subj. to } & z_i \in P_i^\ell. \end{aligned} \tag{7}$$

Consider now the associated column in the form $h_i = [c_i^\top \bar{v}_i \quad \bar{v}_i^\top \quad e_i^\top]^\top$. Then, h_i allows for a cost improvement if it has positive reduced cost, i.e., if $(c_i - \pi)\bar{v}_i - \mu_i > 0$.

iii) A *pivoting* operation is performed, that is, all columns with positive reduced cost are included in the RMP, while columns of the RMP that are not associated to basic variables are dropped[3]. Then, the procedure is iterated until no more columns with positive reduced cost can be found. Let $\Lambda^{\star\ell}$ be the the final optimal solution of the relaxed version of (5) obtained with this procedure. Let $\bar{\lambda}_i^q$ be the entry of $\Lambda^{\star\ell}$ associated to a vertex $v_i^q \in Q_i^\ell$. Then, the solution $z^{\star\ell}$ of $\mathcal{MP}^\ell$ can be expressed as (c.f. (3))

$$z_i(\ell) = \sum_{q=1}^{|Q_i^\ell|} \bar{\lambda}_i^q v_i^q, \quad i = 1, \dots, N. \tag{8}$$

Remark 2.3: When applying the Dantzig-Wolfe Decomposition, we follow the approach in [4] and do not relax the binary constraints in (2). This results in local knapsack problems (7) that can be efficiently solved through dynamic programming schemes, [35]. □

III. Distributed Branch-and-Price Method

In this section, we provide a purely distributed algorithm, inspired by the centralized branch-and-price scheme, to solve (2) in a peer-to-peer network. Then, we provide a cloud-assisted version of the algorithm taking advantage of an aiding computing unit. We assume a solver for Linear Programs is available. In particular, we use the simplex algorithm proposed in [36] to find the unique *lexicographically minimal optimal* solution of a LP and the associated optimal basis.

A. Purely Distributed Branch-and-Price Algorithm

In the proposed distributed algorithm, called *Distributed Branch-and-Price*, each agent i maintains and updates, at the generic time t, local optimal cost and solution candidates J_i^t and $z_{[i]}^t$, as well as a local tree $\mathcal{T}_i^t$. Each agent also maintains and updates a label $\mathcal{L}_i^t$ indicating which problem in $\mathcal{T}_i^t$ it is solving. The

[3]We refer the reader to Appedix A for the definition of basic variables.

candidate optimal solution of a generic problem $\mathcal{MP}_i^\ell$ of $\mathcal{T}_i^t$, for some ℓ, is characterized in terms of a small, representative set of columns called *basis* (c.f. Appendix A). We denote as B_i^t the candidate optimal basis of agent i at time t. At each communication round t, the generic agent i constructs a local restricted master program RMP$_i$, see (9), with the same structure as (6) in which the columns are given by the ones of the bases B_j^t of its neighbors $j \in \mathcal{N}_{i,t}^{\text{in}}$. To streamline the notation, we denote by $\bar{V}_i$ the stack of vertexes v_i^q received by the agent, by $\bar{c}_{V,i}$ the stack of related costs $c_i^\top v_i^q$ and by $\tilde{\Lambda}_i$ the optimization variable. Agent i solves its local RMP$_i$, updates the candidate basis B_i^t, and recovers the associated optimal dual variables $[\pi_i^t \;\; \mu_i^t]$. With the dual solution of the local RMP$_i$ at hands, agent i solves the pricing problem (10) (in the form of (7)) in order to generate, as discussed in Section II, a new column h_i.[4] If such column improves the overall cost, i.e., it has positive reduced cost, agent i substitutes one column of B_i^t with h_i. This is done according to a so called PIVOT operation.

Each time an agent detects convergence, or receives a label $\mathcal{L}_j^t > \mathcal{L}_i^t$ from some neighbor $j \in \mathcal{N}_{i,t}^{\text{in}}$, it sets $\mathcal{L}_i^{t+1} = \mathcal{L}_i^t + 1$. Then, it retrieves the local cost and solution $J_i^{\text{LP}}, z_{[i]}^{\text{LP}}$ from B_i^{t+1} through a EXTRACTSOL function. If $J_i^{\text{LP}} \geq J_i^t$ and $z_{[i]}^{\text{LP}} \in \{0,1\}^{NM}$, it updates the local candidate optimal cost and solution as $J_i^{t+1} = J_i^{\text{LP}}$, $z_{[i]}^{t+1} = z_{[i]}^{\text{LP}}$. Otherwise, it sets $J_i^{t+1} = J_i^t$, $z_{[i]}^{t+1} = z_{[i]}^t$. If $J_i^{\text{LP}} \geq J_i^t$ but $z_{[i]}^{\text{LP}} \notin \{0,1\}^{NM}$ it performs a branching operation. We denote by BRANCH the routine that updates $\mathcal{T}_i^t$ according to a branching on $z_{[i]}^{\text{LP}}$. Finally, the agent starts to solve a new problem, if any, by updating, through an EXTRACTCONSTR function, the local constraint set P_i^{t+1}. From now on we assume that the routines BRANCH and EXTRACTCONSTR are common to all the agents. The whole procedure is summarized in Table 1 from the perspective of agent i.

The convergence properties of the Distributed Branch-and-Price algorithm are stated in the next theorem.

Theorem 3.1: Let (2) be feasible and Assumption 2.1 hold. Consider the sequences $\{J_i^t, z_{[i]}^t\}_{t\geq 0}$, $i \in \{1,\dots,N\}$ generated by the Distributed Branch-and-Price algorithm. Then, in a finite number $\overline{T} \in \mathbb{N}$ of communication rounds, agents agree on a common optimal solution $z^\star$ with optimal cost value $J^\star$ of (2), i.e., $J_i^t = J^\star$ and $z_{[i]}^t = z^\star$, $\forall i \in \{1,\dots,N\}$ and $\forall t \geq \overline{T}$. □

We refer the reader to Appendix B for the proof of Theorem 3.1.

We discuss some interesting features of the proposed distributed scheme. First, agents do not need to know the universal slotted time t. That is, agents can run the steps of the distributed algorithm according to their own local clock. If an agent is performing its computation it is assumed not to have outgoing edges on the communication graph and the steps are performed accordingly to the available in-neighbor bases. This implies that the proposed distributed scheme works under *asynchronous* communication networks. Second, as it will be shown in the analysis, the i-th agent can detect that convergence to an optimal basis has occurred if its basis B_i^t does not change for $2LN+1$ communication rounds. In this way, it can halt the steps in CASE 1 of Algorithm 1. Third, during the first iterations an agent i may not have enough information to solve the RMP$_i$ (9). Thus, it plugs into the local problem a set of artificial variables, eventually discarded during the evolution of the algorithm, with high cost. This method, also called Big-M method, allows the agents to always find a solution to the RMP$_i$. Finally, we underline that the assumption that (2) is feasible can be relaxed to include unfeasible GAPs, but this assumption allows us to lighten the discussion.

B. A Cloud-Assisted Distributed Branch-And-Price

In this section, we provide a cloud-assisted distributed branch-and-price algorithm. In this scenario, agents harness the communication with a *Cloud* node to speed-up the convergence time and reduce the local memory and computing capability requirements. This computing unity is not involved in the column generation steps. Rather, it is involved in the storage of the branching tree. Thereby, agents do not have to

[4]Here, $P_i^t = \{z_i \in \{0,1\}^M \mid z_i \in P_i, z_i \in \Delta_i^\ell\}$, with Δ_i^ℓ being the set of branching binary constraints associated to the problem $\mathcal{MP}_i^\ell$ that agent i is solving at iteration t.

Algorithm 1 Distributed Branch-and-Price Algorithm

Initialization: $B_i^0 = B_{H_M}$ obtained via big-M, incumbent cost $J_i^0 = -\infty$

Evolution: for all $t = 1, 2, \ldots$

Receive $B_j^t, \mathcal{L}_j^t$ from $j \in \mathcal{N}_{i,t}^{\text{in}}$

CASE 1: For each $j \in \mathcal{N}_{i,t}^{\text{in}}$, $\mathcal{L}_j^t \leq \mathcal{L}_i^t$

Set

$$\begin{bmatrix} \bar{c}_{V,i}^\top \\ \bar{V}_i \end{bmatrix} = \bigcup_{j \in \mathcal{N}_{i,t}^{\text{in}} \bigcup \{i\}} B_j^t.$$

Find optimal basis B_i^{t+1} and dual solution $[\pi_i^t \;\; \mu_i^t]$ of

$$\begin{aligned} \max_{\tilde{\Lambda}_i} \;& \bar{c}_{V,i}^\top \tilde{\Lambda}_i \\ \text{subj. to } & \bar{V}_i \tilde{\Lambda}_i = 1_M, \\ & (1_N 1_{|\tilde{\Lambda}_i|}^\top) \tilde{\Lambda}_i = 1_N, \\ & \tilde{\Lambda}_i \geq 0_{|\tilde{\Lambda}_i|}. \end{aligned} \tag{9}$$

Generate column h_i solving

$$\begin{aligned} \max_{z_i} \;& (c_i - \pi_i^t)^\top z_i \\ \text{subj. to } & z_i \in P_i^t. \end{aligned} \tag{10}$$

Update $B_i^{t+1} = \text{PIVOT}(B_i^{t+1}, h_i)$

$J_i^{t+1} = J_i^t$, $z_{[i]}^{t+1} = z_{[i]}^t$, $P_i^{t+1} = P_i^t$, $\mathcal{L}_i^{t+1} = \mathcal{L}_i^t$, $\mathcal{T}_i^{t+1} = \mathcal{T}_i^t$

If B_i^{t+1} has not changed for $2NL + 1$ rounds

GOTO CASE 2

CASE 2: There exists $j \in \mathcal{N}_{i,t}^{\text{in}}$ s.t. $\mathcal{L}_j^t > \mathcal{L}_i^t$

$\mathcal{L}_i^{t+1} = \mathcal{L}_i^t + 1$

$z_{[i]}^{\text{LP}}, J_i^{\text{LP}} = \text{EXTRACTSOL}(B_i^{t+1})$

CASE 2.1: $z_{[i]}^{\text{LP}} \in \{0,1\}^{NM}$, $J_i^{\text{LP}} \geq J_i^t$

$J_i^{t+1} = J_i^{\text{LP}}$, $z_{[i]}^{t+1} = z_{[i]}^{\text{LP}}$

CASE 2.2: $z_{[i]}^{\text{LP}} \notin \{0,1\}^{NM}$, $J_i^{\text{LP}} \geq J_i^t$

$\mathcal{T}_i^{t+1} = \text{BRANCH}(\mathcal{T}_i^t, z_{[i]}^{\text{LP}})$

If $\mathcal{T}_i^{t+1}$ is empty

HALT

$P_i^{t+1} = \text{EXTRACTCONSTR}(\mathcal{T}_i^{t+1})$

construct local branching trees. Also, agent data remain private and the number of messages exchanged at each communication round does not increase.

The algorithm follows the same flow of the Distributed Branch-and-Price scheme proposed in Section III-A. An informal representation of the cloud-assisted distributed branch-and-price algorithm evolution is depicted in Figure 2.

When an agent i, at time $\bar{t}_\ell$, detects that convergence to an optimal solution of a problem $\mathcal{MP}^\ell$ has

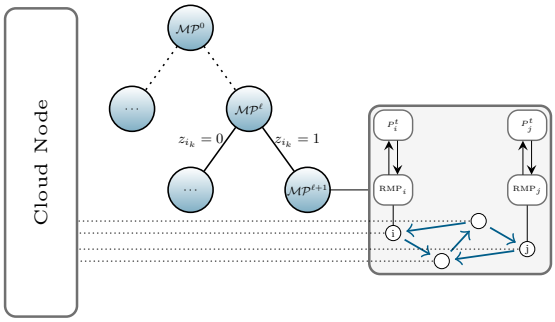


Fig. 2. Evolution of the cloud-assisted distributed branch-and-price algorithm. At iteration t, agents (white circles) are solving problem $\mathcal{MP}^{\ell+1}$ in the tree (blue nodes). The i-th agent solves its RMP$_i$ and generates a new column by solving a pricing problem on P_i^t. Each agent has the possibility to intermittently send/receive info to/from the cloud (dashed lines).

occurred, it sends the basis $B_i^{\bar{t}_\ell}$ to the Cloud. When the cloud node receives such basis, it extracts the optimal cost and solution $J^{\star\ell}$ and $z^{\star\ell}$ and analyzes them according to the steps in CASE 2 of Algorithm 1. Then, if the tree is not empty, the Cloud extracts a new problem $\mathcal{MP}^{\ell+1}$ from the tree according to the extraction strategy. Let $\Delta^{\ell+1}$ be the set of constraints needed to define the sets $P_i^{\ell+1}$ for all $i \in \{1,\dots,N\}$. Let i_k^π, with $\pi \in \{1,\dots,|\Delta^{\ell+1}|\}$, be the index of the variable defining the π-th constraint. Then, the Cloud broadcasts to each agent $i \in \{1,\dots,N\}$ the constraints ($z_{i_k^\pi}=0$ or $z_{i_k^\pi}=1$) that each processor needs to define the local set $P_i^{\ell+1}$. The proof of the cloud-based version follows similar arguments as the one of Theorem 3.1 and is thus omitted.

IV. NUMERICAL COMPUTATIONS

In order to assess the performance and highlight the main features of our distributed algorithm, we provide a set of numerical computations. In the following, we generate new problems, during the branching procedure, by adding constraints in the form $z_{i_k}=0$ and $z_{i_k}=1$, where z_{i_k} is the first non-integer entry of the vector z. Regarding the order in which problems are extracted and solved, we adopt the widely used *depth first* selection procedure, [35]. In this approach, the generated problems are stored in a *stack*, thereby the extraction procedure follows a LIFO approach. Each time a branching occurs, the new problems are placed on the top of the stack. In our implementation, we insert in the first position the problem in which $z_{i_k}=0$ is added at last.

We perform Monte Carlo simulations on random GAP instances. We generate such instances according to four different random models, usually referred to as Model A, B, C and D, of increasing difficulty. We refer the reader to [2] for a survey on such models. Let $\mathcal{U}(a,b)$ denote the discrete uniform distribution on the interval $[a,b]$. The data are generated as follows.

- *Model A:* $w_{\ell m}^A \in \mathcal{U}(10,25)$, $p_{\ell m}^A \in \mathcal{U}(5,25)$ and $g_\ell^A = 9(M/N) + 0.4\max_{1\le\ell\le N}\sum_{m\in\mathcal{J}_\ell^\star} w_{\ell m}$, with $\mathcal{J}_\ell^\star := \{m \mid \ell = \operatorname{argmin}_r p_{rm}\}$.
- *Model B:* $w_{\ell m}^B = w_{\ell m}^A$, $p_{\ell m}^B = p_{\ell m}^A$ and $g_\ell^B = 0.7 g_\ell^A$.
- *Model C:* $w_{\ell m}^C = w_{\ell m}^A$, $p_{\ell m}^C = p_{\ell m}^A$ and $g_\ell^C = \sum_{1\le m\le M} w_{\ell m}/m$.
- *Model D:* $w_{\ell m}^D \in \mathcal{U}(1,100)$, $p_{\ell m}^D = 100 - w_{\ell m} + k$, with $k \in \mathcal{U}(1,21)$ and $g_\ell^D = g_\ell^C$.

We consider different scenarios by varying the number of agents and tasks, thus considering problems with different size and task-over-agents ratio. As for the number of agents, $N = 5, 10, 15$, while, for the number of tasks, $M = 20, 30$.

We generate 20 random instances for each scenario and for each model. We are interested in both time and memory performance of the distributed algorithm. Thus, we show the time that is needed to terminate the algorithm, expressed in terms of the number of communication rounds, and the maximum number of tree nodes stored by the agents. As the problem size increases, the solution of these problems requires the exploration of thousands of tree nodes, see, e.g., [2]. However, in practical scenarios where

assignment problems have to be solved almost in realtime, it is useful to consider a feasible sub-optimal solution to the problem instead of an optimal one. Thereby, even though our algorithm is able to find an optimal solution, in the proposed simulations agents interrupt the distributed algorithm when they find a feasible (sub-optimal) solution. For this reason, we also provide the relative error, in terms of cost value, between the exact solution (evaluated through a centralized solver) and the solution found by the agents. As for the connectivity among agents, we consider a static network modeled by a cyclic digraph. We underline that our algorithm adapts to more complex graph models. However, the choice of such digraph is interesting for simulation purposes due to the fact that it is the static digraph with largest diameter. Thus, the expected number of communication rounds to completion is expected to be higher with respect to graphs with smaller diameter.

The results are shown in Table I, where the results from the Monte Carlo have been averaged over the number of trials of each simulation scenario. We highlight that, in all the simulations, the average relative error is always below 5%. The time to convergence increases with the task-to-agent ratio (M/N). As an example, see Table I, Model A with $N = 15$ and $M = 30$ requires less communication rounds than Model A with $N = 5$ and $M = 30$, even though the overall number of optimization variable is larger. This behavior of the Distributed Branch-and-Price algorithm appears to be consistent with the one reported in the literature for centralized methods. Similarly, Model D is far more difficult to be solved than Model A and requires more communication rounds to be solved, see, for example, the communication rounds needed to solve Model A and Model D with $N = 5, M = 20$. We underline that the number of communication rounds strictly depends on the graph diameter. Since we run the algorithm on a cyclic digraph, which has diameter equal to $N - 1$, the results provided in Table I are the one expected in case of loose connectivity. The maximum number of stored nodes exhibits a similar behavior. That is, as the task-to-agent ratio increases and more difficult models are considered, the distributed algorithm has to explore more branches.

TABLE I
NUMERICAL RESULTS

Model	N	M	Communication Rounds (Avg)	Relative Error (Avg)	Stored Nodes (Avg)
A	5	20	102.95	0.02%	1.35
	5	30	292.05	0%	1.4
	10	20	81.65	0.02%	1.1
	10	30	140.7	0.01%	1.3
	15	20	92.7	0%	1.05
	15	30	120.25	0%	1
B	5	20	227.7	1.09%	3.95
	5	30	511.95	0.26%	4
	10	20	120.05	0.15%	1.85
	10	30	306	0.19%	3.2
	15	20	138.8	0.05%	1.7
	15	30	197.9	0.04%	1.8
C	5	20	192.75	0.5%	3.45
	5	30	648.4	0.65%	5.1
	10	20	180.4	0.75%	3.15
	10	30	473.25	0.41%	5.35
	15	20	163.15	0.25%	2.3
	15	30	466.9	0.43%	4.6
D	5	20	1136.75	4.15%	19.15
	5	30	4018.2	3.84%	31.8
	10	20	600.95	0.87%	9.05
	10	30	5959.95	4.96%	63.55
	15	20	326.95	0.41%	3.9
	15	30	6171.65	3.37%	56.15

V. Experiments on Dynamic GAP for a Team of Ground and Aerial Robots

In the following, we provide experimental results on a dynamic generalized assignment scenario where a team of heterogeneous (ground and aerial) mobile robots has to accomplish a set of tasks that are not completely known in advance. We start by describing the dynamic scenario and we propose the *Distributed Dynamic Assignment and Servicing Strategy*, a resolution methodology for the dynamic task assignment scenario. Then, we discuss how we implemented such strategy into the ROS framework and provide experiments on a real fleet of ground and aerial robots.

A. Distributed Dynamic Assignemnt: Scenario and Strategy

The scenario evolves as follows. We consider a team of ground and aerial mobile robots moving in a three-dimensional environment parametrized by a frame $\{x, y, z\}$. Robots are "smart" cyber-physical agents endowed with communication, computation and actuation capabilities. A set of tasks, parametrized by a position on the $\{x, y\}$ plane, are scattered in the environment. Some of the tasks can be accomplished only by ground robots, other are accessible only to aerial robots and there are tasks that can be performed by all the robots. The generic task is considered accomplished if the robot designed to perform it visits the task location and stands still for a certain random time T^{H}. As in practical applications, the information about the problem instance is not known in advance and new data arrive while the agents are fulfilling other tasks. To adapt the Distributed Branch-and-Price algorithm to such dynamic scenario, we combine it with the methodology proposed in [23] into an optimization and task-fulfilling approach which we call Distributed Dynamic Assignment and Servicing Strategy. Such procedure combines a distributed optimization phase with a planning and control scheme to steer the robots over the assigned tasks. More in detail, the experiment starts with the cyber-physical agents running the cloud version of the Distributed Branch-and-Price Algorithm on a set of tasks known in advance. The cost values p_{im} in (1) represent the travel time for the agent to reach the task. This is evaluated as the robot-task distance (on the $\{x, y\}$ plane) scaled by the robot maximum speed (1 m/s for the UAVs and 0.22 m/s for the ground vehicles). The fact that a task m is not accessible to a certain robot i is modeled by taking $w_{im} > g_i$ in (1). In the following, we assume that the sets P_i, generated randomly according to Model A in Section IV, are fixed throughout the scenario evolution. As soon as an optimal allocation has been found, each robot evaluates the shortest path connecting the tasks it has to perform, by solving a Shortest Hamiltonian Path Problem (SHPP), and starts moving. As soon as a robot reaches the designed task, it stands still on the location for a random time T^{H} between 3 and 5 seconds. In the proposed experiment, we consider a dynamic scenario in which the number of tasks appearing during the evolution is always smaller than the number of served tasks. For the sake of simplicity, we suppose that one new task is made available to robots each time a task has been fulfilled. In this way, the size of the optimization problem is constant. We point out that the strategy can be applied to more general cases where more tasks are revealed. As soon as new tasks appear, the cyber-physical agents run the Distributed Branch-and-Price algorithm on a problem including the new tasks and discarding the visited ones. Specifically, the cost vector entries change according to agents new positions. Meanwhile, each robot keeps performing tasks according to its latest allocation. An example of the evolution of this strategy is displayed in Figure 3.

B. Experimental ROS Architecture

Next, we introduce the experimental implementation of the Distributed Dynamic Assignment and Servicing Strategy with the ROS framework. To carry out our experiments, we used a fleet of 3 Crazyflie nano-quadrotors and 2 Turtlebot Burger. We underline however that the architecture description is independent of the number and kind of robots. Each cyber-physical agent consists of three *ROS nodes*, namely *Optimization*, *Planner* and *Controller* ROS nodes, see Figure 4. It is worth noticing that, in general, each agent has a dedicated machine on which these processes run, so that there is no need for

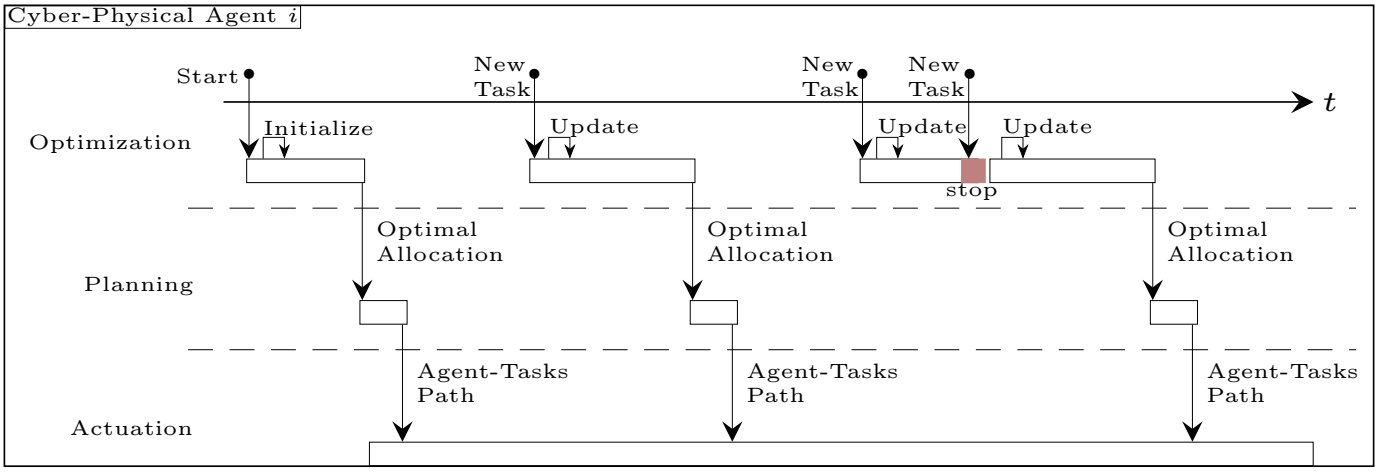


Fig. 3. An example of the Distributed Dynamic Assignment and Servicing Strategy evolution from the perspective of the generic cyber-physical agent. Each time a new task appears, the robot updates the local problem data and re-start the optimization. If a new task arrives during the re-optimization, this latter is halted (red rectangle) and a new one starts. When robot-to-task paths are evaluated, robot actuation changes accordingly.

a central computing unit handling the agents. The Optimization node handles the steps of the distributed optimization algorithm of the associated cyber-physical agent. It communicates with the Optimization nodes of the other robots through the ROS *publisher-subscriber* communication protocol according to a fixed communication graph, and exchange messages containing the local candidate bases. Note that the communication among processes in ROS is completely *asynchronous*. As shown in the theory this is handled by our distributed algorithm. Each time such process receives a message from a neighbor, a *callback function* stores the received basis. Each node performs an iteration of the Distributed Branch-and-Price algorithm within a *loop* of 5 ms. At the beginning of this loop, the node performs one step of the column generation algorithm with the received bases. Then, it sends the updated basis to its neighbors and stays idle until the next loop iteration. We implement another node modeling the Cloud that handles the tree and triggers the agents if a new task has appeared. Optimization and Cloud ROS nodes characterize the *Optimization Layer* (c.f. Figure 4) of the proposed architecture. The Control and Planner ROS nodes constitute instead the *Control Layer* of the proposed software. More in detail, the Planner node generates, through polynomial splines, a sufficiently smooth trajectory connecting all the tasks allocated to a certain robot. The Controller implements a trajectory tracking strategy. It receives the pose of the vehicle by a Vicon motion capture system and sends the control inputs to the robot actuators (*Physical Layer* in Figure 4). In Figure 5, a snapshot from the experiment representing one of the cyber-physical agents solving a GAP instance. Figure 6, taken from the experiment, depicts the moment in which a task has been accomplished, a new one appears and robots update the local problem data. In Figure 7 we show a scenario in which the new optimization significantly changes the allocations of the robots. A video from an experiment is available as supplementary material to the manuscript.

VI. Conclusion

In this paper, we proposed a purely distributed branch-and-price approach to solve the Generalized Assignment Problem in a network of agents, endowed with computation and communication capabilities, that are aware of only a small part of the global optimization problem data. Agents cooperatively solve a relaxations of the GAP by means of a distributed column generation algorithm, targeted for this particular scenario involving binary optimization variables. Since the solution of this relaxation may not be feasible for the GAP, agents cooperatively generate and solve new optimization problems, considering each time additional constraints. Also, we provided a cloud-assisted version of the distributed algorithm, which accelerates the process of generating new problems and requires less memory and computation capabilities

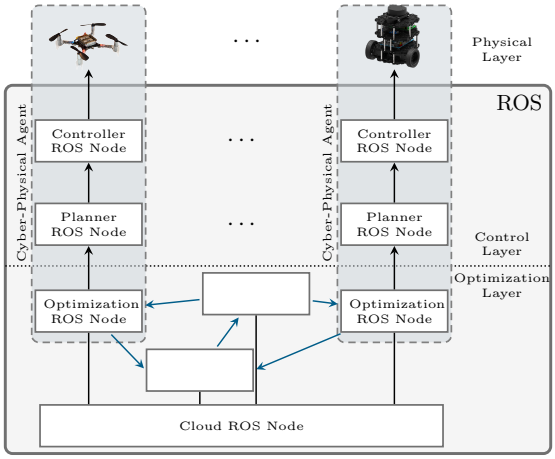


Fig. 4. Distributed Dynamic Assignment and Servicing Strategy architecture. Blue rectangles represent the smart cyber-physical agents endowed with computation, communication and actuation capabilities.

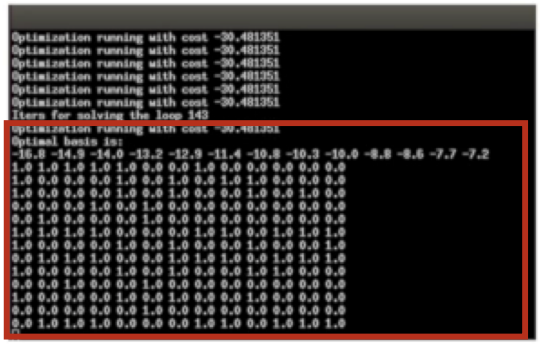


Fig. 5. Snapshot from an experiment. One of the Optimization ROS nodes. In the red rectangle, the optimal basis found by the robot.

from the network agents. Finally, we considered a task assignment scenario where tasks appear dynamically during time. We implemented the proposed algorithms in a ROS based testbed and showed results from experiment on a team of ground and aerial vehicles executing the dynamic task assignment.

ACKNOWLEDGMENT

The authors would like to thank Alessandro Rucco for the fruitful discussions and Nicola Mimmo for the support during the experiments.

APPENDIX A
LINEAR PROGRAMS

An LP in *standard form* is a problem in the form

$$\begin{aligned} \min_{x} \ & c^\top x \\ \text{subj. to } \ & Ax = b, \\ & x \geq 0. \end{aligned} \tag{11}$$

where $c \in \mathbb{R}^d$, $A \in \mathbb{R}^{r \times d}$ and $b \in \mathbb{R}^r$ are the problem data and $x \in \mathbb{R}^d$ is the optimization variable. All the problem constraints are expressed as equality constraints and the variables must be non-negative. A *column* for the problem in (11) is a vector in the form $[c_\ell \ \ A_\ell^\top] \in \mathbb{R}^{r+1}$ where $A_\ell^\top$ is the ℓ-th column of A. A *basis* B is a set of r independent columns of the LP. We denote by c_B (A_B) the sub-vector

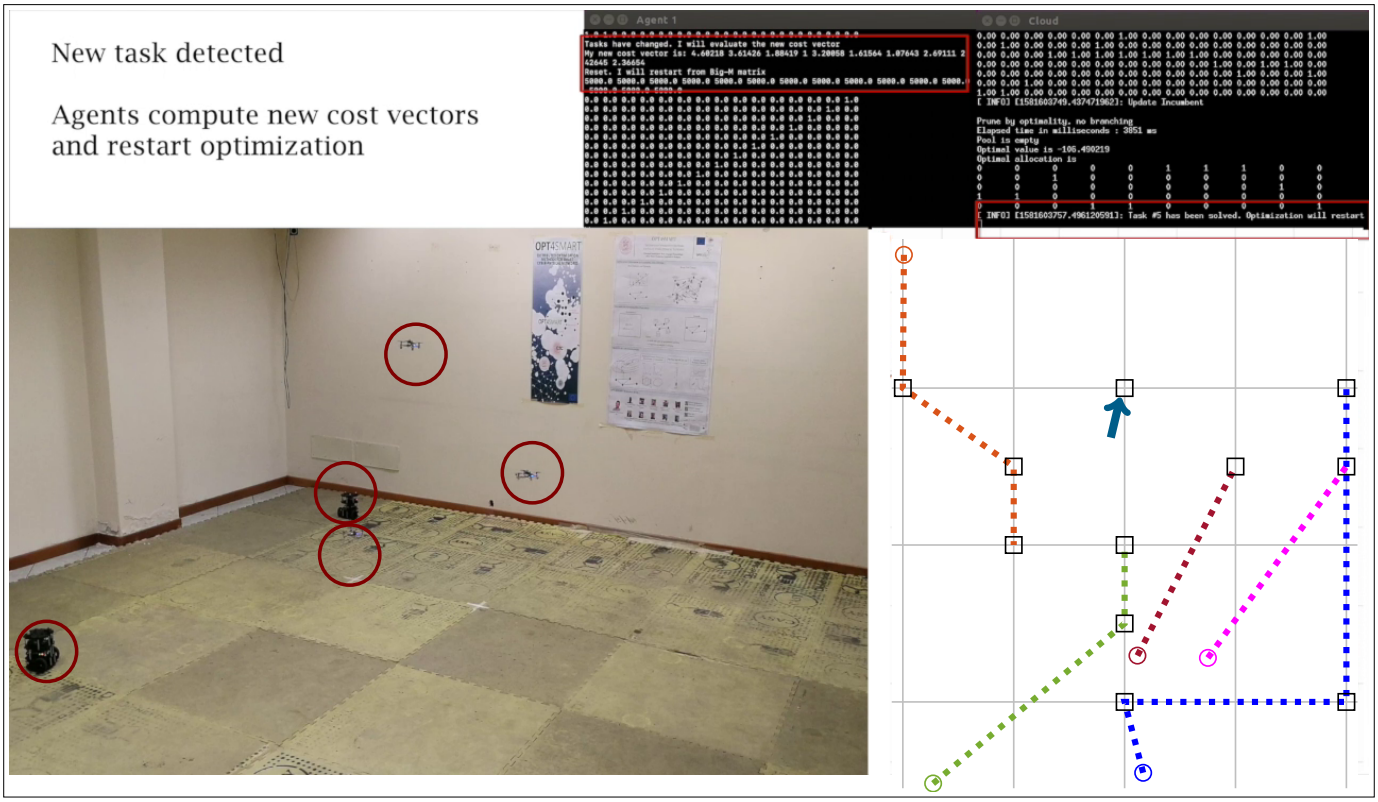


Fig. 6. Snapshot from an experiment. A new task has appeared (indicated by the arrow). The Cloud detects the new task (right terminal). Left terminal represents one of the robots updating its problem data after the task detection.

(sub-matrix) of c (A) constructed from the columns in B. Assume that a solution $x^\star$ to (11) exists. Then, it can be shown that $x^\star$ can be decomposed into two sub-vectors $x^\star_B \neq 0$ of *basic* variables and $x^\star_N = 0$ of *non-basic* variables. A basis represents a *minimal representation* of a linear program, i.e., it is a subset of the problem data representing the problem solution. It can be shown that there exists a basis B such that $x^\star_B$ is the solution of:

$$\begin{aligned} \min_x \ & c_B^\top x \\ \text{subj. to } & A_B x = b, \\ & x \geq 0. \end{aligned}$$

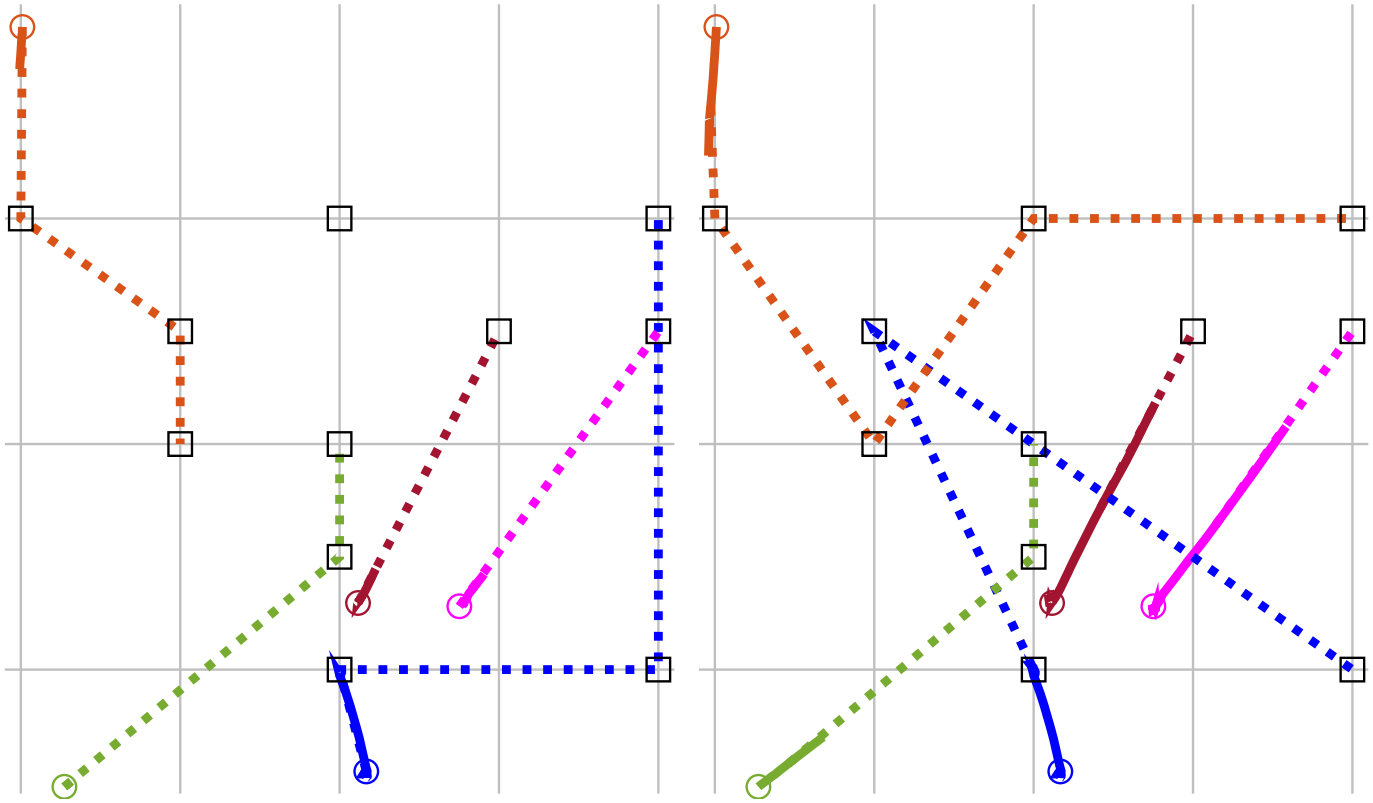
Appendix B
Proof of Theorem 3.1

In order to prove the statement, we show that there exists a monotonically increasing time sequence $\{\bar{t}_\ell\}_{\ell\in\{0,\dots,\ell_{\text{end}}\}}$, for some $\ell_{\text{end}} \in \mathbb{N}$, such that, at each $\bar{t}_\ell$:

i) For all $i, j \in \{1, \dots, N\}$, $\mathcal{T}_i^{\bar{t}_\ell} = \mathcal{T}_j^{\bar{t}_\ell}$, $\mathcal{MP}_i^\ell = \mathcal{MP}_j^\ell = \mathcal{MP}^\ell$, $\mathcal{L}_i^{\bar{t}_\ell} = \mathcal{L}_j^{\bar{t}_\ell} = \ell$ and there exists some $i \in \{1, \dots, N\}$ such that $\mathcal{L}_i^{\bar{t}_\ell - 1} \neq \ell$;

ii) in a finite number of communication rounds, at a time $\bar{t}_{\ell+1} \leq \bar{t}_\ell + Q^\ell$, $Q^\ell \in \mathbb{N}$, either $\mathcal{MP}_i^{\ell+1} = \mathcal{MP}_j^{\ell+1}$ (with $\mathcal{T}_i^{\bar{t}_{\ell+1}} = \mathcal{T}_j^{\bar{t}_{\ell+1}}$ and $\mathcal{L}_i^{\bar{t}_{\ell+1}} = \mathcal{L}_j^{\bar{t}_{\ell+1}}$, $\forall i, j \in \{1, \dots, N\}$) or agents halt the distributed algorithm, i.e., $\ell = \ell_{\text{end}}$, with $J_i^{\bar{t}_{\ell+1}} = J^\star$ and $z_{[i]}^{\bar{t}_{\ell+1}} = z^\star$ optimal cost and solution of (2) for all $i \in \{1, \dots, N\}$.

First notice that i) holds trivially at $t_0 = 0$, since all the agents start solving the relaxed version of (5), namely $\mathcal{MP}^0$, and each agents initializes $\mathcal{L}_i^0 = 0$. Now, we assume that i) holds for some ℓ and prove that ii) holds. We show that, in at most $Q^\ell \in \mathbb{N}$ communication rounds, they reach consensus on an optimal



(a) Trajectories before reoptimization

(b) Trajectories after reoptimization

Fig. 7. The figure depicts how robot allocations change when a new task is detected. Solid lines represent robot positions as the scenario evolves, dotted lines are the desired references connecting the tasks. In (a), a new task has appeared. The cyber-physical agents restart the optimization procedure and find a new set of allocations. In (b), the trajectories including the new task.

solution $\Lambda^{\star\ell}$ of $\mathcal{MP}^\ell$. First, we show that $\mathcal{MP}^\ell$ can be interpreted as an optimization problem obtained by applying the Dantzig-Wolfe decomposition to the following Linear Program:

$$\begin{aligned} \max_{z_1,\dots,z_N} \quad & \sum_{i=1}^{N} c_i^\top z_i \\ \text{subj. to} \quad & \sum_{i=1}^{N} z_i = 1_M, \\ & z_i \in \text{conv}(P_i^\ell), i = 1,\dots,N. \end{aligned} \tag{12}$$

Indeed, we recall that, for GAPs, the vertexes of $\text{conv}(P_i^\ell)$ coincide with the points $v_i^q, q \in Q_i^\ell$, [4]. Thus, points $z_i \in \text{conv}(P_i^\ell)$ can be represented as $z_i = \sum_{q=1}^{|Q_i^\ell|} v_i^q \lambda_i^q$ with $\sum_{q=1}^{|Q_i^\ell|} \lambda_i^q = 1$ and $\lambda_i^q \geq 0$. Let Λ be the

stack of the variables λ_i^q. By substituting these equations in (12) one obtains a problem in the form

$$\begin{aligned} \max_{\Lambda} \ & \sum_{i=1}^{N} \sum_{q=1}^{|Q_i^\ell|} (c_i^\top v_i^q) \lambda_i^q \\ \text{subj. to } & \sum_{i=1}^{N} \sum_{q=1}^{|Q_i^\ell|} v_i^q \lambda_i^q = 1_M, \\ & \sum_{q=1}^{|Q_i^\ell|} \lambda_i^q = 1, i = 1, \dots, N, \\ & \lambda_i^q \geq 0, \end{aligned}$$

that is, the problem $\mathcal{MP}^\ell$.

Notice that the resulting pricing problem is

$$\begin{aligned} \max \ & (c_i - \pi)^\top z_i \\ \text{subj. to } \ & z_i \in \text{conv}(P_i^\ell). \end{aligned} \tag{13}$$

By definition of convex hull and linearity of the cost function, (13) shares the same optimal vertexes of (7).

Thereby, the steps of CASE 1 in Algorithm 1 can be seen as applied to LP (12). By applying the arguments in [22, Theorem 4.4], it can be shown that agents reach consensus, in a finite number of communication rounds $\bar{Q}^\ell$, on a basis B^ℓ corresponding to an optimal solution $\Lambda^{\star\ell}$ of $\mathcal{MP}^\ell$. Moreover, using Assumption 2.1 and arguments similar to the ones in [37, Theorem 9.3], each agent i can halt, at some time $\bar{Q}^\ell \leq t_{i,\ell} \leq \bar{t}_\ell + \bar{Q}^\ell + 2LN + 1$, the steps of CASE 1 if its basis B_i^t has not changed for $2LN + 1$ communication rounds (c.f. Algorithm 1). At these times, each agent obtains the same cost $J^{\star\ell}$ and solution $z^{\star\ell}$ of $\mathcal{MP}^\ell$ (retrieved from $\Lambda^{\star\ell}$ by applying (3)) and sets $\mathcal{L}_i^{t_{i,\ell}+1} = \ell + 1$. If CASE 2.1 in Algorithm 1 occurs, then each agent $i \in \{1, \dots, N\}$ sets $J_i^{t_{i,\ell}+1} = J^{\star\ell}$ and $z_{[i]}^{t_{i,\ell}+1} = z^{\star\ell}$. Instead, if CASE 2.2 occurs, each agent expands the local tree $\mathcal{T}_i^{t_{i,\ell}+1}$. Notice that agents run the BRANCH routine on the same data ($\mathcal{T}_i^{\bar{t}_\ell}$ and $z^{\star\ell}$), so they update the same tree with the same new problems. Finally, if there are still problems to be solved in $\mathcal{T}_i^{t_{i,\ell}+1}$, each agent extracts a new problem $\mathcal{MP}_i^{\ell+1}$. Since the routine EXTRACTCONSTR is common to all the agents and the constructed trees are identical, $\mathcal{MP}_i^{\ell+1} = \mathcal{MP}^{\ell+1}$ for all i. Otherwise, if $\mathcal{T}_i^{t_{i,\ell}+1}$ is empty, each agent halts the Distributed Branch-and-Price Algorithm. Let $Q^\ell = \bar{Q}^\ell + 2LN + 2$ and let $\bar{t}_{\ell+1} = \max_i\{t_{i,\ell}\} + 1$. From the above arguments, $\bar{t}_\ell \leq \bar{t}_{\ell+1} \leq \bar{t}_\ell + Q^\ell$.

Now we show that, if agents halt the distributed algorithm, i.e., $\ell = \ell_{\text{end}}$, then $J_i^{\bar{t}_{\ell+1}} = J^\star$ and $z_{[i]}^{\bar{t}_{\ell+1}} = z^\star$ for all $i \in \{1, \dots, N\}$, with $J^\star$ and $z^\star$ optimal cost and solution of (2). First, notice that $J_i^{\bar{t}_{\ell+1}}$ and $z_{[i]}^{\bar{t}_{\ell+1}}$ are the optimal cost value and solution of some problem $\mathcal{MP}^\ell$ such that $J^{\star\ell} \geq J_i^t$ for each $t \leq \bar{t}_{\ell+1}$ and $z^{\star\ell} \in \{0, 1\}^{NM}$. Since (2) is feasible, and in the branch-and-price algorithm all the nodes of the tree are explored (except the ones discarded during the pruning operation) then each agent has run at least one time the steps in CASE 2.1. Thereby, $J_i^{\bar{t}_{\ell+1}}$ must be equal to the optimal cost value $J^\star$ of (2) and, similarly, $z_{[i]}^{\bar{t}_{\ell+1}} = z^\star$ optimal solution to (2). To conclude, we underline that agents can generate only a finite number of problems. Indeed, the number of additional constraints ($z_{i_k} = 0$ and $z_{i_k} = 1$) they can add is at most 2^{NM}. Thus, there exists a time $\overline{T} = \bar{t}^{\ell_{\text{end}}} + Q^{\ell_{\text{end}}}$ in which all the agents must halt the distributed scheme. This concludes the proof.

REFERENCES

[1] T. Öncan, "A survey of the generalized assignment problem and its applications," *INFOR: Information Systems and Operational Research*, vol. 45, no. 3, pp. 123–141, 2007.

[2] M. Savelsbergh, "A branch-and-price algorithm for the generalized assignment problem," *Operations research*, vol. 45, no. 6, pp. 831–841, 1997.

[3] S. Martello and P. Toth, "Generalized assignment problems," in *International Symposium on Algorithms and Computation*. Springer, 1992, pp. 351–369.

[4] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Operations research*, vol. 46, no. 3, pp. 316–329, 1998.

[5] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[6] E. Hartuv, N. Agmon, and S. Kraus, "Scheduling spare drones for persistent task performance under energy constraints," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 532–540.

[7] J. Bellingham, M. Tillerson, A. Richards, and J. P. How, "Multi-task allocation and path planning for cooperating UAVs," in *Cooperative control: models, applications and algorithms*. Springer, 2003, pp. 23–41.

[8] M. C. Gombolay, R. J. Wilcox, and J. A. Shah, "Fast scheduling of robot teams performing tasks with temporospatial constraints," *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 220–239, 2018.

[9] M. Turpin, N. Michael, and V. Kumar, "An approximation algorithm for time optimal multi-robot routing," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 627–640.

[10] T. Shima, S. Rasmussen, and D. Gross, "Assigning micro UAVs to task tours in an urban terrain," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 4, pp. 601–612, 2007.

[11] D. P. Bertsekas, "The auction algorithm: A distributed relaxation method for the assignment problem," *Annals of operations research*, vol. 14, no. 1, pp. 105–123, 1988.

[12] M. B. Dias, B. Kannan, B. Browning, E. Jones, B. Argall, M. F. Dias, M. Zinck, M. Veloso, and A. Stentz, "Sliding autonomy for peer-to-peer human-robot teams," in *Proceedings of the international conference on intelligent autonomous systems*, 2008, pp. 332–341.

[13] D. A. Castanón and C. Wu, "Distributed algorithms for dynamic reassignment," in *IEEE Conference on Decision and Control (CDC)*, vol. 1, 2003, pp. 13–18.

[14] K. Lerman, C. Jones, A. Galstyan, and M. J. Matarić, "Analysis of dynamic task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 25, no. 3, pp. 225–241, 2006.

[15] M. Alighanbari and J. P. How, "Decentralized task assignment for unmanned aerial vehicles," in *IEEE Conference on Decision and Control (CDC)*, 2005, pp. 5668–5673.

[16] C. Nam and D. A. Shell, "Robots in the huddle: Upfront computation to reduce global communication at run time in multirobot task allocation," *IEEE Transactions on Robotics*, 2019.

[17] M. Hassan, D. Liu, S. Huang, and G. Dissanayake, "Task oriented area partitioning and allocation for optimal operation of multiple industrial robots in unstructured environments," in *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*. IEEE, 2014, pp. 1184–1189.

[18] N. Karapetyan, J. Moulton, J. S. Lewis, A. Q. Li, J. M. O'Kane, and I. Rekleitis, "Multi-robot dubins coverage with autonomous surface vehicles," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2373–2379.

[19] S. Chopra, G. Notarstefano, M. Rice, and M. Egerstedt, "A distributed version of the hungarian method for multirobot assignment," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 932–947, 2017.

[20] M. Bürger, G. Notarstefano, F. Bullo, and F. Allgöwer, "A distributed simplex algorithm for degenerate linear programs and multi-agent assignments," *Automatica*, vol. 48, no. 9, pp. 2298–2304, 2012.

[21] A. Settimi and L. Pallottino, "A subgradient based algorithm for distributed task assignment for heterogeneous mobile robots," in *IEEE Conference on Decision and Control (CDC)*, 2013, pp. 3665–3670.

[22] M. Bürger, G. Notarstefano, and F. Allgöwer, "Locally constrained decision making via two-stage distributed simplex," in *IEEE Conferece on Decision and Control and European Control Conference (CDC-ECC)*, 2011, pp. 5911–5916.

[23] S. Karaman and G. Inalhan, "Large-scale task/target assignment for UAV fleets using a distributed branch and price optimization scheme," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 13 310–13 317, 2008.

[24] V. Pilloni, M. Franceschelli, L. Atzori, and A. Giua, "Deployment of applications in wireless sensor networks: a gossip-based lifetime maximization approach," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 5, pp. 1828–1836, 2016.

[25] A. Kwok and S. Martinez, "A distributed deterministic annealing algorithm for limited-range sensor coverage," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 4, pp. 792–804, 2011.

[26] L. Abbatecola, M. P. Fanti, G. Pedroncelli, and W. Ukovich, "A distributed cluster-based approach for pick-up services," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 960–971, 2018.

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

[28] L. Luo, N. Chakraborty, and K. Sycara, "Distributed algorithm design for multi-robot generalized task assignment problem," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 4765–4771.

[29] ——, "Distributed algorithms for multirobot task assignment with task deadline constraints," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 876–888, 2015.

[30] R. K. Williams, A. Gasparri, and G. Ulivi, "Decentralized matroid optimization for topology constraints in multi-robot allocation problems," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 293–300.

[31] N. Buckman, H.-L. Choi, and J. P. How, "Partial replanning for decentralized dynamic task allocation," in *AIAA Scitech 2019 Forum*, 2019, p. 0915.

[32] Z. Talebpour and A. Martinoli, "Adaptive risk-based replanning for human-aware multi-robot task allocation with local perception," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3790–3797, 2019.

[33] F. Vanderbeck, "On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm," *Operations Research*, vol. 48, no. 1, pp. 111–128, 2000.
[34] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem," *Operations research*, vol. 9, no. 6, pp. 849–859, 1961.
[35] S. Martello, "Knapsack problems: algorithms and computer implementations," *Wiley-Interscience series in discrete mathematics and optimization*, 1990.
[36] C. N. Jones, E. C. Kerrigan, and J. M. Maciejowski, "Lexicographic perturbation for multiparametric linear programming with applications to control," *Automatica*, vol. 43, no. 10, pp. 1808–1816, 2007.
[37] J. M. Hendrickx and V. Blondel, "Graphs and networks for the analysis of autonomous agent systems." Ph.D. dissertation, Catholic University of Louvain, Louvain-la-Neuve, Belgium, 2008.

Andrea Testa received the Laurea degree "summa cum laude" in Computer Engineering from the Università del Salento, Lecce, Italy in 2016.

He is a Research Fellow at Alma Mater Studiorum Università di Bologna, Bologna, Italy. He has been pursuing a Ph.D. in Engineering of Complex Systems at the Università del Salento, Lecce, Italy since November 2016. He was a visiting scholar at LAAS-CNRS, Toulouse, (July to September 2015 and February 2016) and at Alma Mater Studiorum Università di Bologna (October 2018 to June 2019).

His research interests include control of UAVs and distributed optimization.

Giuseppe Notarstefano received the Laurea degree summa cum laude in electronics engineering from the Università di Pisa, Pisa, Italy, in 2003 and the Ph.D. degree in automation and operation research from the Università di Padova, Padua, Italy, in 2007.

He is a Professor with the Department of Electrical, Electronic, and Information Engineering G. Marconi, Alma Mater Studiorum Università di Bologna, Bologna, Italy. He was Associate Professor (from June 2016 to June 2018) and previously Assistant Professor, Ricercatore (from February 2007), with the Università del Salento, Lecce, Italy. He has been Visiting Scholar at the University of Stuttgart, University of California Santa Barbara, Santa Barbara, CA, USA and University of Colorado Boulder, Boulder, CO, USA. His research interests include distributed optimization, cooperative control in complex networks, applied nonlinear optimal control, and trajectory optimization and maneuvering of aerial and car vehicles.

Dr. Notarstefano serves as an Associate Editor for *IEEE Transactions on Automatic Control*, *IEEE Transactions on Control Systems Technology*, and *IEEE Control Systems Letters*. He has been also part of the Conference Editorial Board of IEEE Control Systems Society and EUCA. He is recipient of an ERC Starting Grant 2014.