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

Andrea Testa, *Member, IEEE*, and Giuseppe Notarstefano, *Member, IEEE*

***Abstract*—In this article, 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. We prove finite-time convergence of the algorithm to an optimal solution of the problem. Then, we apply the proposed scheme to a generalized assignment scenario, in which a team of robots has to serve a set of tasks. We implement the proposed algorithm in a Robot Operating System testbed and provide experiments for a team of heterogeneous robots solving the assignment problem.**

***Index Terms*—Cooperating Robots, distributed robot systems, optimization and optimal control, planning, scheduling and coordination.**

I. Introduction

THE *generalized assignment problem* (GAP) is a well-known combinatorial optimization problem with several applications, such as vehicle routing, facility location, resource scheduling, and supply chain, to name a few [1]–[3]. Even though the GAP is an NP-hard problem, several approaches have been developed for solving this problem for both exact and approximate solutions. We refer the reader to [4] for a survey. Branch-and-price algorithms [2], [5] are among the most investigated algorithms allowing for both optimal and suboptimal solutions.

A. Related Work

Task assignment naturally arises in cooperative robotics, where heterogeneous agents collaborate to fulfill a complex task; see, e.g., [6] for an early reference. Specific applications include persistent monitoring of locations [7], path planning of mobile robots, e.g., unmanned aerial vehicle (UAVs) [8], task scheduling for robots working in the same space [9], vehicle routing [10], and task assignment in urban environments [11]. 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 [12]. A market-based approach is considered in [13] for the coordination of human–robot teams. Castanón and C. Wu [14] propose an algorithm, based on a sequential shortest augmenting path scheme, to solve a dynamic multitask allocation problem. Agents propose assignments that are validated by a coordinating unit. As for decentralized schemes, Lerman *et al.* [15] solve a dynamic task allocation problem for robots that can perform local sensing operations and do not communicate with each other. In [16], a task assignment problem is solved, in a decentralized scheme, through the so-called petal algorithm. In [17], 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 multirobot systems is proposed in [18] and solved by a genetic algorithm. An area coverage problem in marine environments is solved in [19] with heuristics based on the traveling salesman problem.

As for distributed schemes, i.e., with processors in a peer-to-peer network without a central coordinator, a distributed version of the Hungarian method is proposed in [20]. A distributed simplex scheme for degenerate linear programs (LPs) is proposed in [21] in the context of multiagent assignment problems. A distributed subgradient is applied in [22] to a task assignment problem, while in [23], a distributed column generation scheme is proposed. A linear task assignment problem with time-varying cost functions is considered in [24], while an optimal role and position assignment problem is addressed in [25] by iteratively solving a sequence of linear assignment problems. In these approaches, the 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 [26], a large-scale distributed task/target

Manuscript received May 26, 2021; revised September 18, 2021; accepted October 4, 2021. Date of publication November 4, 2021; date of current version June 7, 2022. This work was supported by the European Research Council under the European Union's Horizon 2020 Research and Innovation Program under Grant 638992—OPT4SMART. This article was recommended for publication by Associate Editor N. Agmon and Editor P. Robuffo Giordano upon evaluation of the reviewers' comments. *(Corresponding author: Andrea Testa.)*

The authors are with the Department of Electrical, Electronic and Information Engineering, University of Bologna, 40136 Bologna, Italy (e-mail: a.testa@unibo.it; giuseppe.notarstefano@unibo.it).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TRO.2021.3120046.

Digital Object Identifier 10.1109/TRO.2021.3120046

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

assignment problem across a fleet of autonomous UAVs is considered, but the communication graph is assumed to be complete. In the context of wireless sensor networks, Pilloni *et al.* [27] propose a distributed task allocation in order to maximize the network lifetime. A distributed task assignment algorithm is used in conjunction with a deterministic annealing in [28], in the context of limited-range sensor coverage. In [29], a dynamic vehicle routing problem (VRP) is approached with a distributed protocol, in which agents iteratively solve graph partitioning problems. In the works [30], [31], Testa *et al.* address mixed-integer LPs by means of a distributed cutting-plane algorithm and apply it to a multiagent multitask assignment problem. Distributed implementations of the auction-based algorithm are often used to solve task assignment problems (see, e.g., [32] for an early reference) and in particular GAPs [33], [34]. The auction-based approach allows for a suboptimal solution with performance guarantees. In [35], this approach is applied to a task allocation problem expressed as a combinatorial optimization problem with matroid constraints. In the recent works [36], [37], a dynamic task allocation scenario with partial replanning is considered. The above references show that the exact resolution of the GAP is an open problem in a purely distributed setting. Indeed, state-of-the-art solutions are usually based on proper linear relaxations or suboptimal approaches.

B. Contributions

In this article, we propose a purely distributed version of the branch-and-price algorithm to solve the GAP 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. With respect to the aforementioned works, and specifically [26], [32], and [33], the proposed scheme has the following distinctive new features. To the best of the authors' knowledge, this is the first attempt to solve the GAP to optimality in a purely distributed fashion. Remarkably, the proposed scheme is shown to also converge under time-varying and directed communication networks. Moreover, it is worth noticing that the approaches in [30] and [31] are not applicable to the considered GAP scenario, which involves equality constraints. Finally, we apply the proposed algorithm to a dynamic task assignment problem, where tasks may arrive during time and robots have to adapt the local plan according to the new information. An experimental platform, based on the Robot Operating System (ROS), is proposed to run experiments, in which a team of aerial and ground robots cooperatively solve the GAP relying on the proposed distributed branch-and-price scheme.

The rest of this article is organized as follows. In Section II, we introduce the distributed setup considered throughout this article. Then, we introduce the GAP and a centralized scheme, called *branch-and-price*, to solve it. In Section III, we propose a purely distributed branch-and-price algorithm. In Section IV, we provide numerical simulations for randomly generated GAPs. In Section V, we show the results of experiments on a swarm of heterogeneous robots. Finally, Section VI concludes this article.

Notation: We denote by e_ℓ the ℓ-th vector of the canonical basis (e.g., $e_1 = [1\,0\,\ldots\,0]^\top$) 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 GAP addressed in this article. Also, the (centralized) branch-and-price scheme is illustrated.

A. Distributed Problem Setup

In the GAP, 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 ($x_{im} = 0$). We denote constraints in the form $x_{im} \in \{0, 1\}$ as integer constraint. Then, the standard integer programming formulation is

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

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 $z_i = [x_{i1},\ldots,x_{iM}]^\top \in \mathbb{R}^M, \forall i = 1,\ldots,N$. In the following, we denote by z the stack $[z_1^\top,\ldots,z_N^\top]^\top$. Also, let $c_i = [p_{i1},\ldots,p_{iM}]^\top \in \mathbb{R}^M$, $D_i = [w_{i1},\ldots,w_{iM}] \in \mathbb{R}^M$, and $P_i = \{z_i \in \{0,1\}^M \mid D_i z_i \le g_i\}$, for $i = 1,\ldots,N$. Then, (1) can be recast as

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

This new formulation of (1) allows us to point out the distributed nature of the problem. Namely, c_i describes the profits associated with assigning tasks to agent i, $\sum_{i=1}^{N} z_i = 1_M$ describes the

assignment constraints (*coupling constraints*), and 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 us 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. We refer the reader to [5] for a more detailed dissertation. For the sake of clarity, we organize this subsection in three parts.

1) Dantzig–Wolfe Decomposition for GAPs: An equivalent formulation of (2), which is exploited in our distributed setup, can be obtained as follows. Such a procedure, originally introduced in [5], is strictly related to the *Dantzig–Wolfe decomposition* [38]. 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

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

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, [5]. Let

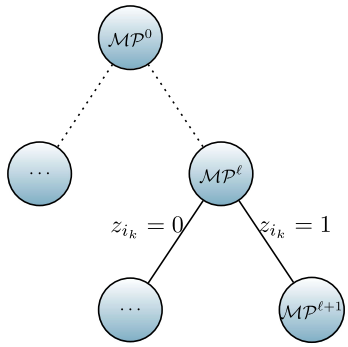


Fig. 1. Example of branching tree with problems generated according to constraints on the sets P_i.

$\Lambda \in \mathbb{R}^{\sum_{i=1}^N |Q_i|}$ be the stack of all the combiners. Substituting (3) and (4) into (2) leads to the following equivalent integer programming master problem:

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

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].

2) 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 [5] explores the set of feasible solutions of the 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., Fig. 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 a 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, respectively. Branches (edges) indicate the constraints that have to be 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 that 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 *lower bound*, 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.

1) *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 $z^{\text{INC}} = z^{\star\ell}$.
2) *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.
3) *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).

3) 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 [39] in the context of cutting stock problems). It consists in iteratively performing the following three steps.

1) A *restricted master problem* (RMP) is solved, made by a small subset of the *columns*[2] of $\mathcal{MP}^\ell$ with sets $\bar{Q}_i^\ell \subset Q_i^\ell$ and a smaller combiner vector $\tilde{\Lambda}$, i.e.,

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

2) 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 with 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 with the constraint $\sum_{q \in \bar{Q}_i^\ell} \tilde{\lambda}_i^q = 1$. For each $i = 1, \ldots, N$, a new column is found by solving the so-called *pricing problem:*

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

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

3) 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 with 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 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 with a vertex $v_i^q \in Q_i^\ell$. Then, the solution $z^{\star\ell}$ of $\mathcal{MP}^\ell$ can be expressed as [cf. (3)]

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

Remark 2.3: When applying the Dantzig–Wolfe decomposition, we follow the approach in [5] 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 [40]. □

We collect all the relevant symbols in Table I.

TABLE I
LIST OF THE MAIN SYMBOLS AND THEIR DEFINITIONS

Standard GAP Formulation	
$N \in \mathbb{N}_{>0}$	Number of robots
$M \in \mathbb{N}_{>0}$	Number of tasks
$x_{im} \in \{0,1\}$	1 if robot i serves task, 0 otherwise
$z_i \in \mathbb{R}^M$	$[x_{i1}, \ldots, x_{iM}]^\top$
$p_{im} \in \mathbb{R}_{\geq 0}$	Reward if robot i serves task m
$c_i \in \mathbb{R}^M$	$[p_{i1}, \ldots, p_{iM}]^\top$
$g_i \in \mathbb{R}_{\geq 0}$	Capacity of robot i
$w_{im} \in \mathbb{R}_{\geq 0}$	Capacity consumption of task m for robot i
$D_i \in \mathbb{R}^M$	$[w_{i1}, \ldots, w_{iM}]$
$P_i \subseteq \{0,1\}^M$	$\{z_i \in \{0,1\}^M \mid D_i z_i \leq g_i\}$
Dantzig-Wolfe Reformulation	
Q_i	set of extreme points of P_i
$v_i^q \in \{0,1\}^M$	q-th extreme point of P_i
$\lambda_i^q \in \{0,1\}$	Combiner associated to v_i^q
$\Lambda \in \mathbb{R}^{\sum_{i=1}^N \lvert Q_i \rvert}$	Stack of combiners
Branch-and-Price	
$\mathcal{MP}^\ell$	ℓ-th node of the branching tree
$z^{\star\ell}, J^{\star\ell}$	Optimal solution and cost of $\mathcal{MP}^\ell$
P_i^ℓ	Constraint set of robot i at node $\mathcal{MP}^\ell$
z^{INC}, $J^{\text{INC}} = J^{\star\ell}$	Candidate GAP solution and cost

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. We assume that a solver for LPs is available. In particular, we use the simplex algorithm proposed in [41] to find the unique *lexicographically minimal optimal* solution of an LP and the associated optimal basis.

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 candidate optimal solution of a generic problem

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

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

$\mathcal{MP}_i^\ell$ of $\mathcal{T}_i^t$, for some ℓ, is characterized in terms of a small representative set of columns called *basis* (cf. Appendix A). We denote by 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 in the form

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

Notice that this problem has the same structure as (6), where the columns are the ones of the bases B_j^t with $j \in \mathcal{N}_{i,t}^{\text{in}}$. 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 hand, agent i solves a pricing problem

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

which has the same structure as (7). As discussed in Section II, this allows agents to generate a new column h_i.[4] If such a 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 Algorithm 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

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 $t = 1, 2, \dots$
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

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

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

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

Generate column h_i solving

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

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

[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 with the problem $\mathcal{MP}_i^\ell$ that agent i is solving at iteration t.

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$_i$. As for the communication overhead, at each communication round, each robot sends to its neighbors a matrix of size $(N+M+1)\times(N+M)$. Each column of this matrix is in the form $h_i=[c_i^\top \bar{v}_i \;\; \bar{v}_i^\top \;\; e_i^\top]^\top$. Here, $c_i^\top \bar{v}_i$ is a real number specifying the cost to execute an allocation $\bar{v}_i \in \{0,1\}^M$. The vector $e_i^\top \in \{0,1\}^N$ specifies which robot generated that allocation. It is worth noting that the vector $\bar{v}_i$ can be encoded as an array of M bits, while $e_i^\top$ can be encoded as an integer number. 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.

Remark 3.2: As a possible variation, agents may harness the communication with a *Cloud* node to speed up the convergence time and reduce the local memory and computing requirements. In this architecture, the cloud unity is only involved in the storage of the branching tree (and not in the column generation steps). Thereby, agents do not construct local branching trees. Also, agent data remain private, and the number of messages exchanged at each communication round does not increase. When an agent i, at time $\bar{t}_\ell$, detects that convergence to an optimal solution of a problem $\mathcal{MP}^\ell$ has occurred, it sends the basis $B_i^{\bar{t}_\ell}$ to the cloud. At this point, the cloud 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. Finally, if the tree is not empty, it extracts a new problem $\mathcal{MP}^{\ell+1}$ from the tree according to the extraction strategy and broadcasts to each agent i the additional constraints to build up $P_i^{\ell+1}$. The proof of the cloud-based version follows similar arguments as the one of Theorem 3.1 and is 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. Simulations have been implemented on the DISROPT [42] toolbox and carried out on a laptop equipped with a 2.5-GHz dual core processor and 16 GB of RAM. 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 noninteger entry of the vector z. Regarding the order in which problems are extracted and solved, we adopt the widely used *depth first* selection procedure [40]. In this approach, the generated problems are stored in a *stack*, thereby the extraction procedure follows a last in first out (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 Models 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.

1) *Model A:* $w_{\ell m}^A \in \mathcal{U}(10,25)$, $p_{\ell m}^A \in \mathcal{U}(5,25)$, and $g_\ell^A = 9(M/N)+0.4\max_{1\le\ell\le N}\Sigma_{m\in\mathcal{J}_\ell^\star} w_{\ell m}$, with $\mathcal{J}_\ell^\star := \{m \mid \ell = \operatorname{argmin}_r p_{rm}\}$.
2) *Model B:* $w_{\ell m}^B = w_{\ell m}^A$, $p_{\ell m}^B = p_{\ell m}^A$, and $g_\ell^B = 0.7 g_\ell^A$.
3) *Model C:* $w_{\ell m}^C = w_{\ell m}^A$, $p_{\ell m}^C = p_{\ell m}^A$, and $g_\ell^C = \Sigma_{1\le m\le M} w_{\ell m}/m$.
4) *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 50 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. We also show the equivalent time, in seconds, needed for each simulation. Since DISROPT exploits the message passing interface (MPI) protocol to simulate the agents, the computation time per agent is evaluated as $T_{\text{ag}} = T_{\text{el}} N_{\text{co}}/N$, where T_{el} is the total elapsed time and N_{co} is the number of cores. 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 real time, it is useful to consider a feasible suboptimal 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 (suboptimal) 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 the largest diameter. Thus, the expected number of communication rounds to completion is expected to be higher with respect to graphs with smaller diameter.

The mean value and the standard deviation (evaluated over the number of trials) for each simulation scenario are shown in Table II. 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 II), 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 variables 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

TABLE II
NUMERICAL RESULTS

Model	N	M	Communication Rounds (Avg–Std)	Relative Error (Avg–Std)	Stored Nodes (Avg–Std)	Time (Avg–Std)
A	5	20	83.30–25.36	0.00%–0.00%	1.10–0.36	1.94–0.60
	5	30	329.38–151.87	0.01%–0.08%	1.44–0.64	12.85–5.89
	10	20	75.34–43.86	0.00%–0.00%	1.30–0.78	3.34–1.86
	10	30	107.92–87.52	0.01%–0.07%	1.30–1.06	10.29–7.88
	15	20	76.60–26.23	0.01%–0.05%	1.12–0.38	5.09–2.01
	15	30	95.86–33.27	0.00%–0.00%	1.08–0.34	14.59–4.80
B	5	20	192.04–205.30	1.06%–2.10%	3.50–3.79	4.67–5.03
	5	30	774.50–680.58	0.57%–0.91%	5.04–4.40	33.68–29.46
	10	20	161.36–222.29	0.25%–0.80%	3.14–4.38	6.56–9.13
	10	30	236.36–324.55	0.20%–0.53%	3.24–4.53	23.42–31.24
	15	20	90.02–42.64	0.02%–0.15%	1.36–0.66	5.16–2.43
	15	30	178.40–174.70	0.04%–0.10%	2.16–2.13	27.32–26.81
C	5	20	158.24–149.94	0.63%–1.25%	3.00–3.03	3.95–3.74
	5	30	652.32–741.28	0.48%–1.20%	4.48–5.59	27.98–31.95
	10	20	155.06–230.47	0.47%–1.58%	3.24–4.93	5.81–8.44
	10	30	375.52–431.19	0.59%–1.03%	5.50–6.41	35.37–40.32
	15	20	107.08–133.53	0.14%–0.57%	1.80–2.26	4.85–6.16
	15	30	294.02–306.51	0.24%–0.42%	3.94–4.15	38.92–39.99
D	5	20	1072.76–601.48	4.31%–3.88%	20.10–11.23	33.47–18.10
	5	30	4805.44–2525.97	4.91%–3.65%	36.80–18.05	201.56–101.85
	10	20	933.76–979.26	2.77%–4.12%	18.44–20.06	36.20–36.08
	10	30	5959.95–3596.71	4.96%–4.26%	63.55–38.69	560.14–338.02
	15	20	187.88–164.20	0.22%–0.44%	2.84–2.52	9.38–8.19
	15	30	6171.65–4015	3.37%–2.81%	56.15–37.02	740.5–481.8

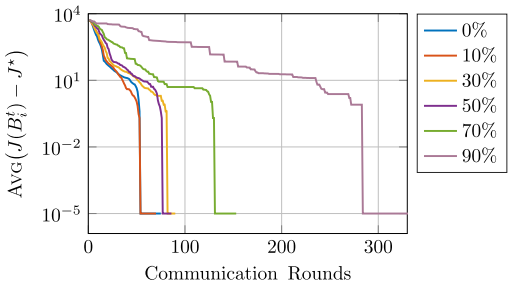


Fig. 2. Cost error during the evolution of the algorithm with different percentages of packet loss.

more difficult to be solved than Model A and requires more communication rounds (see, e.g, the communication rounds needed to solve Models A and 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, whose diameter is $N - 1$, the results provided in Table II are the ones 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. To conclude, we propose a numerical simulation in which robots communicate in a network subject to packet loss. We consider a scenario with $N = 5$ and $M = 20$. Problem data are generated according to Model A. We consider the cases with loss probability 0% (no packet loss), 10%, 30%, 50%, 70%, and 90%. Specifically, at each iteration, the i-th robot discards the message from the jth robot according to the given probability. Results are given in Fig. 2. We show the mean error between the cost $J(B_i^t)$ associated with the basis B_i^t and the optimal solution $J^\star$.

Remark 4.1: Other distributed approaches suitable for the GAP solution are the ones in [32] and [33]. As for the one in [32], the authors consider the case in which $w_{im} \in \{0, 1\}$ for each $i \in \{1, \ldots, N\}$ and for each $m \in \{1, \ldots, M\}$ [cf. (1)]. When the cost function is linear, as in the GAP scenario, the constraint matrix is said to be totally unimodular, and the problem can be solved as a linear problem instead of a mixed-integer problem. Thereby, the first solution found by our algorithm, which is also tailored for general GAPs with nonunimodular structure, is always the optimal one. The one found by the scheme in [32] is guaranteed to be at most 50% suboptimal. Moreover, our algorithm allows for directed communication graphs, while the one in [32] assumes undirected communications. Finally, agents in [32] exchange, at each communication round, two real vectors of size N and M, respectively, and a vector of size M representing which agent is performing each task. As for the distributed approach in [33], each agent has to flood its local variables to all the other agents. This results in multihop communications at each iteration. Moreover, the scheme in [33] is based on the assumption of static undirected graphs. Each agent in [33] sends to its neighbors a real vector of size M and three integers. Similarly to our approach, it considers the solution of a knapsack problem at each iteration. Finally, as in [32], it guarantees at most 50% suboptimality of the solution found. We perform a comparison between the proposed approach and the one in [33] for the scenario with $N = 15$ and $M = 30$. The results are shown in Table III. We took the same underlying communication graph for both the schemes. Since the algorithm in [33] needs agents to flood their information to all the other agents at each communication round, we multiply the total number of iterations of the algorithm by Nd, with d being the diameter of the graph. Besides the problems generated via Model D, our algorithm is able to find in less iterations a solution with

TABLE III
PERFORMANCE COMPARISON

Model	M	N	Distributed Branch-and-Price		[33]	
			Comm. Rounds (Avg–Std)	Rel. Error (Avg–Std)	Comm. Rounds (Avg–Std)	Rel. Error (Avg–Std)
A	15	30	95.86–33.27	0.00%–0.00%	411.6–62.44	0.56%–0.37%
B	15	30	178.40–174.70	0.04%–0.10%	447.3–65.77	1.91%–1.13%
C	15	30	294.02–306.51	0.24%–0.42%	453.6–107.9	3.45%–1.74%
D	15	30	6171.65–4015	3.37%–2.81%	302.4–34.12	0.0%–0.0%

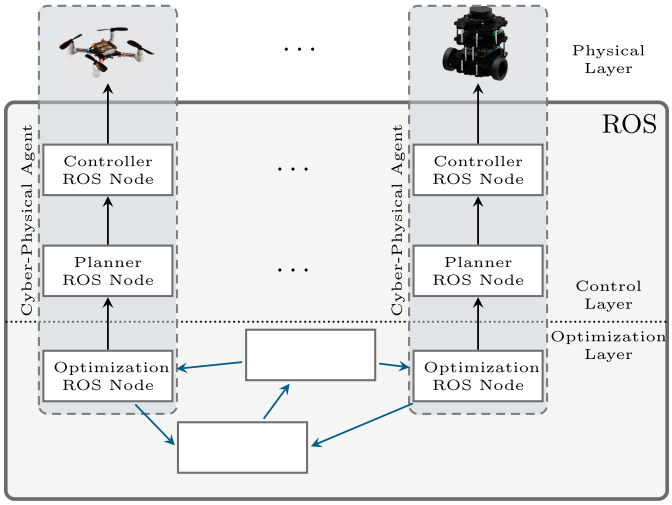


Fig. 3. Distributed dynamic assignment and servicing strategy architecture. Blue rectangles represent the smart cyber-physical agents endowed with computation, communication, and actuation capabilities.

a smaller relative error with respect to the one found by the algorithm in [33]. □

V. EXPERIMENTS ON GAPS FOR A TEAM OF GROUND AND AERIAL ROBOTS

In the following, we provide experimental results on a generalized assignment scenario, where a team of heterogeneous (ground and aerial) mobile robots has to accomplish a set of tasks that may not be completely known in advance. We start by describing how we implemented the proposed distributed scheme into the ROS framework. Then, we propose the *distributed dynamic assignment and servicing strategy*, a resolution methodology for this assignment scenario, and provide experiments on a real fleet of ground and aerial robots.

A. Experimental ROS Architecture

In the proposed architecture, robots are "smart" cyber-physical agents endowed with communication, computation and actuation capabilities. Each cyber-physical agent consists of three *ROS nodes*, namely, *optimization*, *planner*, and *controller* ROS nodes (see Fig. 3). 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 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 exchanges 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 a 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. The optimization nodes characterize the *optimization layer* (cf. Fig. 3) 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 steering a robot over its designated tasks. 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 Fig. 3).

B. Distributed Dynamic Assignment: Scenario and Strategy

The scenario evolves as follows. We consider a team of ground and aerial mobile robots moving in a 3-D environment parameterized by a frame $\{x, y, z\}$. A set of tasks, parameterized 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. For a task to be accomplished, a robot has to visit the task location, stand still for a certain random time T^{H}, and go back to a given depot (e.g., to recharge batteries). 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 a dynamic scenario, we combine it with the methodology proposed in [26] into an optimization and task-fulfilling approach, which we call distributed dynamic assignment and servicing strategy. Such a 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 distributed branch-and-price algorithm on a set of tasks known in advance. Inspired by [32], we pick p_{im} in (1) as a *time-discounted reward*, i.e., $p_{im} = \lambda_m^{\tau_i^m}$, where $\lambda_m \in (0, 1)$ is a scoring value for task m and τ_i^m is the time needed by agent i to reach task m. The time τ_i^m 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

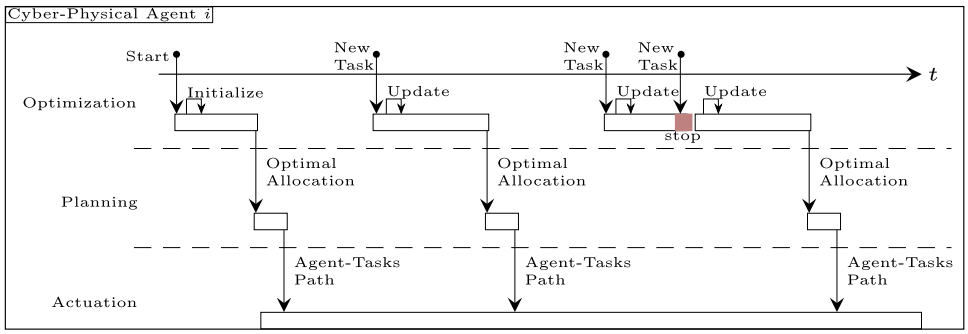


Fig. 4. Example of the distributed dynamic assignment and servicing strategy evolution from the perspective of the generic cyber-physical agent. Each time a new task appears, the robot updates the local problem data and restarts the optimization. If a new task arrives during the reoptimization, this latter is halted (red rectangle) and a new one starts. When robot-to-task paths are evaluated, robot actuation changes accordingly.

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 a robot reaches the designed task, it stands still on the location for a random time T^{H} between 3 and 5 s. 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. Moreover, while in the current setup, we consider the immediate strategy in which we reoptimize the entire problem, one could think of implementing tailored schemes leveraging the dynamic structure of the problem. 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 the new task positions. Meanwhile, each robot keeps performing tasks according to its latest allocation. An example of the evolution of this strategy is shown in Fig. 4. A snapshot from an experiment with two Crazyflie nano-quadrotors and three Turtlebot3 Burger is shown in Fig. 5. Here, robots have terminated the distributed optimization procedure, and one of the allocations is shown. A video is available as supplementary material to this article.[5]

Remark 5.1: As discussed, e.g., in [3], GAPs can also be used to find approximate solutions of VRPs. In general, VRPs penalize the order of execution of the tasks and involve a larger number of variables with respect to GAPs. The idea in [3] is to construct a GAP instance based on the VRP problem data. As soon as a GAP solution has been found, robots perform their associated task in an order that minimizes, e.g., the total traveled distance. This can be done, e.g., by solving a shortest Hamiltonian path problem (SHPP). The proposed distributed dynamic assignment and servicing strategy could be, thus, modified in order to address such scenarios. Specifically, robots start solving the GAP

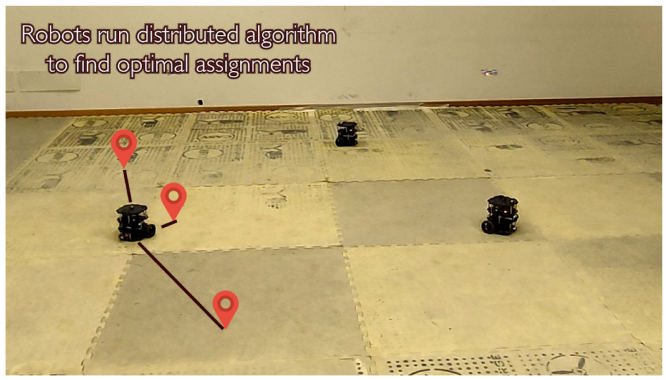


Fig. 5. Snapshot from an experiment. The figure depicts the optimal assignment for one of the robots.

with the available tasks and, once an optimal solution has been found, construct robot-to-tasks paths by solving SHPPs. When a new task arrives, robot resolve the optimization problem and adjust the path according to the new problem data. We performed an experiment with three Crazyflie nano-quadrotors and two Turtlebot3 Burger with the cloud-based approach. A video is available as supplementary material to this article.[6]

VI. Conclusion

In this article, we proposed a purely distributed branch-and-price approach to solve the GAP in a network of agents, endowed with computation and communication capabilities, which 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. Finally, we considered an assignment scenario, where tasks may appear dynamically during time. We implemented the proposed algorithms in a ROS-based testbed and showed results from experiments on a team of ground and aerial vehicles executing

[5]The video is also available at [Online]. Available: https://youtu.be/Sl_3ZmJvvbU.

[6]The video is also available at [Online]. Available: https://youtu.be/vBSJsduFYKQ.

the generalized assignment. Future investigations may include the solution of dynamic instances of the GAP with tailored distributed approaches that do not need to reoptimize the entire problem when new data arrive.

APPENDIX A
LINEAR PROGRAMS

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

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

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 nonnegative. 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 subvector (submatrix) 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 subvectors $x_B^\star \neq 0$ of *basic* variables and $x_N^\star = 0$ of *nonbasic* variables. A basis represents a *minimal representation* of an LP, 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 $x_B^\star$ is the solution of

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

APPENDIX B
PROOF OF THEOREM 3.1

A. Preliminary Lemmas for the Proof of Theorem 3.1

Before proceeding with the proof of Theorem 3.1, we provide two lemmas, which are useful for the analysis.

Lemma B.1: Let Assumption 2.1 hold. Consider a network of N agents running the steps in CASE 1 of Algorithm 1 to solve a node $\mathcal{MP}^\ell$ of the tree. Then, in a finite number of iterations, agents reach consensus to an optimal basis B^ℓ associated with the optimal cost $J^{\star\ell}$ of $\mathcal{MP}^\ell$.

Proof: The proof mimics the one proposed in [23]. We refer the reader to this work for additional details. First, we show that $\mathcal{MP}^\ell$ can be obtained by applying the Dantzig–Wolfe decomposition to the following LP:

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

Indeed, we recall that, for GAPs, the vertexes of $\text{conv}(P_i^\ell)$ coincide with the points $v_i^q, q \in Q_i^\ell$, [5]. 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 into (12), one obtains a problem in the form

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

which is problem $\mathcal{MP}^\ell$. Notice that the resulting pricing problem for each agent i is

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

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). At this point, we note that, during the algorithmic evolution, agent i can update its local candidate basis B_i^t by considering new columns in the local LP. It is worth noting that, starting from any basis B_i^t, there exists a finite number of pivoting operations to the optimal basis B^ℓ. These columns can be found in two ways: 1) by the local column generation routine and the subsequent pivoting and 2) when collecting all the in-neighbors matrices B_j^t with $j \in \mathcal{N}_{i,t}^{\text{in}}$. If $J_i^t < J^{\star\ell}$, there always exists an agent j able to generate a column improving the cost J_i^t after a pivoting. Since the network is connected if that column is fundamental for the evolution of the algorithm, e.g., it belongs to the optimal basis, then agent j will generate it (and include it) in its basis within a finite number of communication rounds. Thus, as soon as $J_i^t < J^{\star\ell}$, there always exists a finite time T_D such that $J_i^t < J_I^{t+T_D}$. Since there exists only a finite number of columns, in a finite number of communication rounds T_f, it stands that $J_i^{T_f} = J^{\star\ell}$ for each i. If a lexicographic solver is considered, then it stands that $B_i^t = B^\ell$ for each i. This concludes the proof.

Lemma B.2: Let Assumption 2.1 hold. Then, a processor has computed its final basis and can halt the execution of the steps in CASE 1 of Algorithm 1 as soon as the value of B_i^t has not changed after $L(2N-1)$ communication rounds.

Proof: Assume that a certain node i satisfies $B_i^t = B^\star, J_i^t = J^\star$ for all $t \in \{t_0,\dots,t_0 + 2L(N-1)\}$ and pick any other node j. Without loss of generality, consider $t_0 = 0$. By L-strong connectivity, after at most L communication rounds, agent i has been able to spread its basis at least to another agent. We now define the set $\bar{\mathcal{N}}_0$ of agents $k \in \{1,\dots N\}$ such that there exists an increasing sequence of time instants $\tau_0,\dots,\tau_m$

comprised between 0 and L (i.e., with $0 \leq \tau_0$ and $\tau_m \leq L$), such that the edges $(i, \ell_1), \ldots, (\ell_m, k)$ belong to the digraph at times $\tau_0, \ldots, \tau_m$. This set is not empty, since the union graph is strongly connected in $[0, L]$. Then, it stands $J_k^L \geq J_i^t, \forall k \in \bar{\mathcal{N}}_0$. Now, consider the interval $[L, 2L]$, for which we define a set similar to $\bar{\mathcal{N}}_0$, but with paths originating from the agents in $\bar{\mathcal{N}}_0 \cup \{i\}$. Formally, consider the set $\bar{\mathcal{N}}_1$ of agents $k \in \{1, \ldots N\}$ such that there exists an increasing sequence of time instants $\tau_0, \ldots, \tau_m$ comprised between L and $2L$ (i.e., with $L \leq \tau_0$ and $\tau_m \leq 2L$), such that the edges $(h, \ell_1), \ldots, (\ell_m, k)$ belong to the digraph at times $\tau_0, \ldots, \tau_m$, for some $h \in \bar{\mathcal{N}}_0 \cup \{i\}$. Notice that $\bar{\mathcal{N}}_0 \subset \bar{\mathcal{N}}_1$, so that $\bar{\mathcal{N}}_1$ has a larger cardinality than $\bar{\mathcal{N}}_0$. Otherwise, the graph would not be strongly connected in $[L, 2L]$. Then, it stands $J_k^{2L} \geq J_i^t$, $\forall k \in \bar{\mathcal{N}}_1$. Iterating at most $N-1$ times, we see that the sets $\bar{\mathcal{N}}_0 \ldots, \bar{\mathcal{N}}_{N-2}$ become larger and larger, so that $j \in \bar{\mathcal{N}}_{N-2}$. Thus, it stands that $J_k^{(N-1)L} \geq J_i^t$. That is, after $(N-1)L$ communication rounds, all the agents have at least the same cost agent i had at time 0. By repeating the same arguments for the converse path, we conclude that $J_i^{2(N-1)L} \geq J_k^{(N-1)L}$. But, by assumption, $J_i^{2(N-1)L} = J^\star$, so that we conclude $J^\star \leq J_j^{(N-1)L} \leq J^\star$, i.e., $J_j^{(N-1)L} = J^\star$. Thus, if B_i^t does not change for $L(2N-1)$ time instants, then its value will never change afterward because all bases $B_j^t, j \in \{1, \ldots, N\}$, have cost equal to $J^\star$ at least as early as time equals LN. □

B. Proof of Theorem 3.1

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

i) for all $i, j \in \{1, \ldots, N\}$, $\mathcal{T}_i^{\bar{t}_\ell} = \mathcal{T}_j^{\bar{t}_\ell}$, $\mathcal{MP}_i^\ell = \mathcal{MP}_j^\ell = \mathcal{MP}^\ell$, $\mathcal{L}_i^{\bar{t}_\ell} = \mathcal{L}_j^{\bar{t}_\ell} = \ell$, and there exists some $i \in \{1, \ldots, N\}$ 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 $\mathcal{L}_i^{\bar{t}_{\ell+1}} = \mathcal{L}_j^{\bar{t}_{\ell+1}}$, $\forall i, j \in \{1, \ldots, N\}$) 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 $i \in \{1, \ldots, N\}$.

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 agent initializes $\mathcal{L}_i^0 = 0$. Now, we assume that i) holds for some ℓ and prove that ii) holds. Then, by applying the arguments in Lemma B.1, 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, by Lemma B.2, 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 (cf. 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 $i \in \{1, \ldots, N\}$ 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 $i \in \{1, \ldots, N\}$, 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.

ACKNOWLEDGMENT

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

REFERENCES

[1] T. Öncan, "A survey of the generalized assignment problem and its applications," *Inf. Syst. Oper. Res.*, vol. 45, no. 3, pp. 123–141, 2007.

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

[3] M. L. Fisher and R. Jaikumar, "A generalized assignment heuristic for vehicle routing," *Networks*, vol. 11, no. 2, pp. 109–124, 1981.

[4] S. Martello and P. Toth, "Generalized assignment problems," in *Proc. Int. Symp. Algorithms Comput.*, 1992, pp. 351–369.

[5] 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," *Oper. Res.*, vol. 46, no. 3, pp. 316–329, 1998.

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

[7] E. Hartuv, N. Agmon, and S. Kraus, "Scheduling spare drones for persistent task performance under energy constraints," in *Proc. 17th Int. Conf. Auton. Agents Multiagent Syst.*, 2018, pp. 532–540.

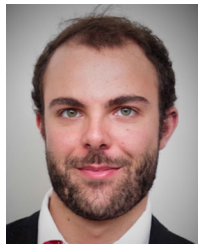
[8] J. Bellingham, M. Tillerson, A. Richards, and J. P. How, "Multi-task allocation and path planning for cooperating UAVs," in *Cooperative Control: Models, Applications and Algorithms*. New York, NY, USA: Springer, 2003, pp. 23–41.

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

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

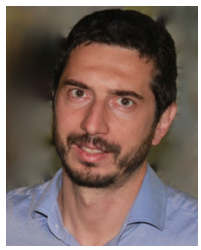
[11] T. Shima, S. Rasmussen, and D. Gross, "Assigning micro UAVs to task tours in an urban terrain," *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 4, pp. 601–612, Jul. 2007.

[12] D. P. Bertsekas, "The auction algorithm: A distributed relaxation method for the assignment problem," *Ann. Oper. Res.*, vol. 14, no. 1, pp. 105–123, 1988.
[13] M. B. Dias *et al.*, "Sliding autonomy for peer-to-peer human-robot teams," in *Proc. Int. Conf. Intell. Auton. Syst.*, 2008, pp. 332–341.
[14] D. A. Castanón and C. Wu, "Distributed algorithms for dynamic reassignment," in *Proc. IEEE Conf. Decis. Control*, 2003, vol. 1, pp. 13–18.
[15] K. Lerman, C. Jones, A. Galstyan, and M. J. Matarić, "Analysis of dynamic task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 25, no. 3, pp. 225–241, 2006.
[16] M. Alighanbari and J. P. How, "Decentralized task assignment for unmanned aerial vehicles," in *Proc. IEEE Conf. Decis. Control*, 2005, pp. 5668–5673.
[17] C. Nam and D. A. Shell, "Robots in the huddle: Upfront computation to reduce global communication at run time in multirobot task allocation," *IEEE Trans. Robot.*, vol. 36, no. 1, pp. 125–141, Feb. 2020.
[18] 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 *Proc. 13th Int. Conf. Control Autom. Robot. Vis.*, 2014, pp. 1184–1189.
[19] 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 *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 2373–2379.
[20] S. Chopra, G. Notarstefano, M. Rice, and M. Egerstedt, "A distributed version of the hungarian method for multirobot assignment," *IEEE Trans. Robot.*, vol. 33, no. 4, pp. 932–947, Aug. 2017.
[21] 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.
[22] A. Settimi and L. Pallottino, "A subgradient based algorithm for distributed task assignment for heterogeneous mobile robots," in *Proc. IEEE Conf. Decis. Control*, 2013, pp. 3665–3670.
[23] M. Bürger, G. Notarstefano, and F. Allgöwer, "Locally constrained decision making via two-stage distributed simplex," in *Proc. IEEE Conf. Decis. Control Eur. Control Conf.*, 2011, pp. 5911–5916.
[24] E. Montijano, D. Tardioli, and A. R. Mosteo, "Distributed dynamic sensor assignment of multiple mobile targets," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4921–4926.
[25] A. R. Mosteo, E. Montijano, and D. Tardioli, "Optimal role and position assignment in multi-robot freely reachable formations," *Automatica*, vol. 81, pp. 305–313, 2017.
[26] S. Karaman and G. Inalhan, "Large-scale task/target assignment for UAV fleets using a distributed branch and price optimization scheme," *IFAC Proc. Vol.*, vol. 41, no. 2, pp. 13310–13317, 2008.
[27] V. Pilloni, M. Franceschelli, L. Atzori, and A. Giua, "Deployment of applications in wireless sensor networks: A gossip-based lifetime maximization approach," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 5, pp. 1828–1836, Sep. 2016.
[28] A. Kwok and S. Martinez, "A distributed deterministic annealing algorithm for limited-range sensor coverage," *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 4, pp. 792–804, Jul. 2011.
[29] L. Abbatecola, M. P. Fanti, G. Pedroncelli, and W. Ukovich, "A distributed cluster-based approach for pick-up services," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 2, pp. 960–971, Apr. 2019.
[30] A. Testa, A. Rucco, and G. Notarstefano, "A finite-time cutting plane algorithm for distributed mixed integer linear programming," in *Proc. Conf. Decis. Control*, 2017, pp. 3847–3852.
[31] A. Testa, A. Rucco, and G. Notarstefano, "Distributed mixed-integer linear programming via cut generation and constraint exchange," *IEEE Trans. Autom. Control*, vol. 65, no. 4, pp. 1456–1467, Apr. 2020.
[32] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 912–926, Aug. 2009.
[33] L. Luo, N. Chakraborty, and K. Sycara, "Distributed algorithm design for multi-robot generalized task assignment problem," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 4765–4771.
[34] L. Luo, N. Chakraborty, and K. Sycara, "Distributed algorithms for multi-robot task assignment with task deadline constraints," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 3, pp. 876–888, Jul. 2015.
[35] R. K. Williams, A. Gasparri, and G. Ulivi, "Decentralized matroid optimization for topology constraints in multi-robot allocation problems," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 293–300.
[36] N. Buckman, H.-L. Choi, and J. P. How, "Partial replanning for decentralized dynamic task allocation," in *Proc. AIAA Scitech Forum*, 2019, Art. no. AIAA 2019-0915.
[37] Z. Talebpour and A. Martinoli, "Adaptive risk-based replanning for human-aware multi-robot task allocation with local perception," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3790–3797, Oct. 2019.
[38] F. Vanderbeck, "On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm," *Oper. Res.*, vol. 48, no. 1, pp. 111–128, 2000.
[39] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem," *Oper. Res.*, vol. 9, no. 6, pp. 849–859, 1961.
[40] S. Martello, *Knapsack Problems: Algorithms and Computer Implementations* (Wiley-Interscience Series in Discrete Mathematics and Optimization). Hoboken, NJ, USA: Wiley, 1990.
[41] 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.
[42] F. Farina, A. Camisa, A. Testa, I. Notarnicola, and G. Notarstefano, "DISROPT: A Python framework for distributed optimization," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 2666–2671, 2020.



Andrea Testa (Member, IEEE) received the Laurea degree (*summa cum laude*) in computer engineering and the Ph.D. degree (*summa cum laude*) in engineering of complex systems from the Università del Salento, Lecce, Italy, in 2016 and 2020, respectively.

He is currently a Research Fellow with the Alma Mater Studiorum Università di Bologna, Bologna, Italy. He was a Visiting Scholar with the Laboratory for Analysis and Architecture of Systems, Scientific Research National Center, Toulouse, France, from July to September 2015 and February 2016, and with the Alma Mater Studiorum—Università di Bologna, Bologna, from October 2018 to June 2019. His research interests include control of unmanned aerial vehicles and distributed optimization.



Giuseppe Notarstefano (Member, IEEE) 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, Padova, Italy, in 2007.

He is currently a Professor with the Department of Electrical, Electronic, and Information Engineering "Guglielmo Marconi," Alma Mater Studiorum Università di Bologna, Bologna, Italy. He was an Associate Professor from June 2016 to 2018 and an Assistant Professor in February 2007 with the Università del Salento, Lecce, Italy. He was a Visiting Scholar with the University of Stuttgart, Stuttgart, Germany; the University of California Santa Barbara, Santa Barbara, CA, USA; and the 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 is an Associate Editor for IEEE TRANSACTIONS ON AUTOMATIC CONTROL, IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, and IEEE CONTROL SYSTEMS LETTERS. He has also been part of the Conference Editorial Board of the IEEE Control Systems Society and European Control Association. He is Recipient of an European Research Council Starting Grant 2014.