ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

Multi-Robot Pickup and Delivery via Distributed Resource Allocation

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:
Andrea Camisa, Andrea Testa, Giuseppe Notarstefano (2023). Multi-Robot Pickup and Delivery via Distributed Resource Allocation. IEEE TRANSACTIONS ON ROBOTICS, 39(2), 1106-1118 [10.1109/tro.2022.3216801].

Availability:
This version is available at: https://hdl.handle.net/11585/905577 since: 2024-03-07

Published:
DOI: http://doi.org/10.1109/tro.2022.3216801

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)

12 October 2024

This is the final peer-reviewed accepted manuscript of:

A. Camisa, A. Testa and G. Notarstefano, "Multi-Robot Pickup and Delivery via Distributed Resource Allocation," in *IEEE Transactions on Robotics*, vol. 39, no. 2, pp. 1106-1118, April 2023.

The final published version is available online at:

https://doi.org/10.1109/TRO.2022.3216801

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/)

When citing, please refer to the published version.

Multi-Robot Pickup and Delivery via Distributed Resource Allocation

Andrea Camisa*, Andrea Testa*, Giuseppe Notarstefano

***Abstract*—In this paper, we consider a large-scale instance of the classical *Pickup-and-Delivery Vehicle Routing Problem* (PDVRP) that must be solved by a network of mobile cooperating robots. Robots must self-coordinate and self-allocate a set of pickup/delivery tasks while minimizing a given cost figure. This results in a large, challenging Mixed-Integer Linear Problem that must be cooperatively solved without a central coordinator. We propose a distributed algorithm based on a primal decomposition approach that provides a feasible solution to the problem in finite time. An interesting feature of the proposed scheme is that each robot computes only its own block of solution, thereby preserving privacy of sensible information. The algorithm also exhibits attractive scalability properties that guarantee solvability of the problem even in large networks. To the best of our knowledge, this is the first attempt to provide a scalable distributed solution to the problem. The algorithm is first tested through Gazebo simulations on a ROS 2 platform, highlighting the effectiveness of the proposed solution. Finally, experiments on a real testbed with a team of ground and aerial robots are provided.**

***Index Terms*—Distributed Optimization; Distributed Robot Systems; Planning, Scheduling and Coordination; Cooperating Robots**

I. Introduction

The Pickup-and-Delivery Vehicle Routing Problem (PDVRP) is one of the most studied combinatorial optimization problems. The interest is mainly motivated by the practical relevance in real-world applications such as such as battery exchange in robotic networks, [1], pickup and delivery in warehouses, [2], task scheduling [3] and delivery with precedence constraints [4]. The PDVRP is known to be an $\mathcal{NP}$-Hard optimization problem, and can be solved to optimality only for small instances. In a PDVRP, a group of vehicles has to fulfill a set of transportation requests. Requests consist of picking up goods at some locations and delivering them to other locations. The problem then consists of determining minimal length paths such that all the requests are satisfied. To achieve this, we can assign to each location a label "P" (pickup) or "D" (delivery), and then define a graph of all possible paths that can be traveled by vehicles. In Figure 1, we show an example scenario with two vehicles, two pickups and two deliveries. Note that, in order to have a well-defined graph, vehicles must start from an initial node representing the initial position (which may be different for each of them) and have to reach a target node. This can be also "virtual" in the sense that, once the last delivery position has been reached, the vehicle stops there and waits for further instructions. With respect to classical Vehicle Routing Problems, the PDVRP introduces a set of additional variables and constraints that make the problem harder to solve. In particular, precedence constraints must ensure that the pickup of a good is performed before its delivery. Moreover, vehicles have capacity constraints that must be satisfied throughout the mission. These additional constraints are based on additional real variables that are not included in classical routing problems. In order to determine the optimal path, one typically formulate an associated optimization problem and solves it to optimality. Throughout the paper, we consider a general version of this optimization problem for which we propose a distributed algorithm, i.e., an algorithm that robots can run in a peer-to-peer fashion without a central coordinator. In this distributed setting, communication among robots occurs according to a given graph and it is not a design parameter. As the distributed computation paradigm requires agents to share computation instead of data, it is particularly recommended in contexts where the local problem data (such as final assignment of tasks, vehicle capacity, cost of tasks, etc.) has to be maintained private as e.g. in military applications or futuristic smart cities with robots belonging to different users.

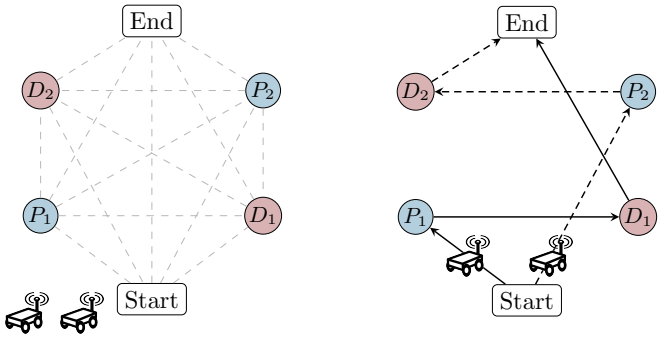

Figure 1. Example PDVRP scenario. Left: vehicles begin from the "start" node and must end at the terminal node. There are two pickup requests P_1 and P_2 and two associated deliveries D_1 and D_2. Right: the optimal path consists of the first vehicle traveling through P_1 and D_1 and the second vehicle traveling through P_2 and D_2.

A. Related Work

Several algorithms have been proposed to solve the problem to (sub)optimality in centralized settings, we refer the reader to the surveys [5]–[8] for a comprehensive list of these methods in static and dynamic settings. Online, dynamic approaches have been proposed, see, e.g., [9], taking into account collision avoidance constraints among robotic agents, [10].

In order to overcome the drawbacks of centralized approaches, as, e.g., the high computational complexity of the

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

* These authors contributed equally to this work.

problem, a branch of literature analyzes schemes based on master-slave or communication-less architectures. In master-slave approaches, implementations of the parallel auction based algorithm are among the most used strategies, see, e.g., [11], [12]. As for communication-less approaches, we refer the reader to [13] for a dynamic pickup and delivery application. In [14] authors address a multi-depot multi-split vehicle routing problem, where computing nodes apply a local heuristic based on a stochastic gradient descent and then exchange the local solutions in order to select the best one.

Few works address the solution of vehicle routing problems in peer-to-peer networks. Indeed, the majority of the approaches in literature address approximate problems or special cases of the pickup-and-delivery. Authors in [15]–[17] solve unimodular task assignment problems by means of convex optimization techniques. Other works are instead based on the solution of mixed-integer problems with a simplified structure. In particular, [18], [19] address generalized assignment problems, where the order of execution of the tasks is not relevant. In [20], authors address an assignment problem where agent-to-task paths are evaluated offline and capacity constraints are not considered. The distributed auction-based approach, see [21], [22] for recent applications, is often used to address task allocation problems. These approaches do not address more complex models with, e.g., load demands and execution time of tasks. As for distributed approaches to vehicle routing problems, in the works [23]–[25], authors propose distributed schemes based on Voronoi partitions for stochastic dynamic vehicle routing problems with time windows and customer impatience. Authors in [26] propose a distributed algorithm where agents communicate according to cyclic graph and perform operations one at a time. In [27], a distributed scheme in which agents iteratively solve graph partitioning problems is proposed, and is shown to converge to a suboptimal solution of a dynamic vehicle routing problem. A multi-depot vehicle routing problem is considered instead in [28], where the problem is modeled as game in which customers and depots find a feasible allocation by means of an auction game.

Since the PDVRP is a Mixed-Integer Linear Program (MILP), let us recall some recent works addressing MILPs in distributed frameworks. A distributed cutting-plane approach converging in finite time with arbitrary precision to a solution of the MILP is proposed in [20]. However, this approach requires each agent to compute the entire solution of the problem and is not suited for multi-robot PDVRPs. The work in [29] addresses large-scale MILPs by a dual decomposition approach, while in [30] the authors propose a primal-decomposition algorithm with low suboptimality bounds. However, to the best of our knowledge, there are no works in the literature specifically tailored for pickup-and-delivery problems.

B. Contributions

The contributions of the present paper are as follows. We formalize the Multi-vehicle Pickup-and-Delivery Vehicle Routing Problem as a large-scale distributed optimization problem with local and coupling constraints. The considered formulation of the Pickup-and-delivery problem is general and encompasses scenarios where each robot can only perform a subset of the tasks. We propose a distributed algorithm for the fast computation of a feasible solution to the problem. The algorithm is based on a distributed resource allocation approach and requires only local computation and communication among neighboring robots with no external coordination. The proposed algorithm is scalable in the sense that the amount of computation performed by each robot is independent of the network size. Moreover, robots do not exchange private information with each other such as vehicle capacity or the computed routes. We formally prove finite-time feasibility of the solution computed by the algorithm and we provide guidelines for practical implementation. We first provide numerical computations on a ROS 2 set-up in which the PDVRP is solved with our algorithm and the dynamics of the vehicles is realistically simulated through Gazebo. Although the problem is $\mathcal{NP}$-hard, the simulations highlight that our distributed algorithm is able to compute "good-quality" solutions within a few (fixed) iterations. Then, we show results of real experiments performed on the ROS 2 testbed with TurtleBot3 ground robots and Crazyflie2 aerial robots. We now highlight the main differences with other related approaches. In [21], [22], authors consider vehicle routing problems with upper-bounds in the number of tasks a robot can serve. Dynamic variations of these set-up are considered in [23]–[25]. Our approach instead takes into account more general capacity constraints. Moreover, we consider precedence constraints among tasks. Authors in [26] propose a scheme for pickup-and-delivery problems in which vehicles execute the steps of the algorithm once at a time and communicate according to a ring graph with a shared token. In our protocol instead agents can perform the optimization steps concurrently, can communicate according to general connected graphs and do not need a shared memory. Authors in [27], [28] consider the set-up in which vehicles start their mission from different depots, pick-up resources at given locations and then come back to their initial depot. In our set-up instead vehicles have to pick-up resources and deliver them to other stations before going to final depots.

The paper is organized as follows. In Section II, the problem statement together with the needed notation and preliminaries is provided. The distributed algorithm is provided in Section III, while Section IV encloses the theoretical analysis. We describe the Gazebo simulations in Section VI and, finally, we provide the experimental results in Section VII.

II. Pickup-and-Delivery Vehicle Routing Problem

In this section, we provide a mathematical formulation of the optimization problem studied in the paper together with a thorough description of its structure.

A. Optimization Problem Formulation

We consider a scenario in which N robots, indexed by $\mathbb{I} := \{1, \dots, N\}$, have to serve the transportation requests. We denote by $P := \{1, \dots, |P|\}$ the index set of pickup requests and by $D := \{|P+1|, \dots, 2|P|\}$ the index set of delivery demands (with $P \cap D = \emptyset$). To each pickup location $j \in P$ is associated a delivery location $j \in D$ (with $|P| = |D|$), so

that both the requests must be served by the same robot. To ease the notation, we also define a set $R := P \cup D$ of *all* the transportation requests (independently of their pickup/delivery nature). Each request $j \in R$ is characterized by a service time $d_j \geq 0$, which is the time needed to perform the pickup or delivery operation. Within each request, it is also associated a load $q_j \in \mathbb{R}$, which is positive if $j \in P$ and negative if $j \in D$. Each robot has a maximum load capacity $C_i \geq 0$ of goods that can be simultaneously held. The travel time needed for the i-th vehicle to move from a location $j \in R$ to another location $k \in R$ is denoted by $t_i^{jk} \geq 0$. In order to travel from two locations j, k, the i-th robot incurs a cost $c_i^{jk} \in \mathbb{R}_{\geq 0}$. Finally, two additional locations s and σ are considered. The first one represents the mission starting point, while the second one is a virtual ending point. For this reason, the corresponding demands q^s, q^σ and service times d^s, d^σ are set to 0.

The goal is to construct minimum cost paths satisfying all the transportation requests. To this end, a graph of all possible paths through the transportation requests is defined as follows. Let $\mathcal{G}_A = (V_A, \mathcal{E}_A)$, be the graph with vertex set $V_A = \{s, \sigma\} \cup R$ and edge set $\mathcal{E}_A = \{(j,k) \mid j,k \in V_A, j \neq k \text{ and } j \neq \sigma, k \neq s\}$. Owing to its definition, $\mathcal{E}_A$ contains edges starting from s or from locations in R and ending in σ or other locations in R. For all edges $(j,k) \in \mathcal{E}_A$, let x_i^{jk} be a binary variable denoting whether vehicle $i \in \{1, \dots, N\}$ is traveling ($x_i^{jk} = 1$) or not ($x_i^{jk} = 0$) from a location j to a location k. Also, let $B_i^j \in \mathbb{R}_{\geq 0}$ be an the optimization variable modeling the time at which vehicle i begins its service at location j. Similarly, let $Q_i^j \in \mathbb{R}_{\geq 0}$ be the load of vehicle i when leaving location j. To keep the notation light, we denote by x the vector stacking x_i^{jk} for all i, j, k and by B, Q the vectors stacking all B_i^j and Q_i^j. The PDVRP can be formulated as the following optimization problem [6],

$$\min_{x,B,Q} \sum_{i=1}^{N} \sum_{(j,k)\in\mathcal{E}_A} c_i^{jk} x_i^{jk} \tag{1a}$$

$$\text{subj. to} \sum_{i=1}^{N} \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq 1 \qquad \forall j \in R \tag{1b}$$

$$\sum_{k:(s,k)\in\mathcal{E}_A} x_i^{sk} = 1 \qquad \forall i \in \mathbb{I} \tag{1c}$$

$$\sum_{j:(j,\sigma)\in\mathcal{E}_A} x_i^{j\sigma} = 1 \qquad \forall i \in \mathbb{I} \tag{1d}$$

$$\sum_{j:(j,k)\in\mathcal{E}_A} x_i^{jk} = \sum_{j:(k,j)\in\mathcal{E}_A} x_i^{kj} \qquad \forall i \in \mathbb{I}, k \in R \tag{1e}$$

$$\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} = \sum_{k:(j+|P|,k)\in\mathcal{E}_A} x_i^{|P|+j,k} \qquad \forall i \in \mathbb{I}, j \in P \tag{1f}$$

$$B_i^j \leq B_i^{j+|P|}, \qquad \forall i \in \mathbb{I}, j \in P \tag{1g}$$

$$x_i^{jk} = 1 \Rightarrow B_i^k \geq B_i^j + d^j + t_i^{jk} \tag{1h}$$

$$x_i^{jk} = 1 \Rightarrow Q_i^k = Q_i^j + q^k \tag{1i}$$

$$\underline{Q}^j \leq Q_i^j \leq \overline{Q}_i^j \qquad \forall j \in V_A, i \in \mathbb{I} \tag{1j}$$

$$Q_i^s = Q_i^{\text{init}} \qquad \forall i \in \mathbb{I} \tag{1k}$$

$$x_i^{jk} \in \{0,1\} \qquad \forall i \in \mathbb{I}, (j,k) \in \mathcal{E}_A, \tag{1l}$$

Table I
LIST OF THE MAIN SYMBOLS AND THEIR DEFINITIONS

Basic definitions	
$N \in \mathbb{N}_{\geq 0}$	Number of vehicles of the system
$\mathbb{I} = \{1, \dots, N\}$	Set of vehicles
P, D	Sets of Pickup and Delivery requests
$R = P \cup D$	Set of all transportation requests
s	Mission starting point
σ	Mission ending point (virtual or physical)
$V_A = \{s, \sigma\} \cup R$	Set of PDVRP graph vertices
$\mathcal{E}_A \subset V_A \times V_A$	Set of PDVRP graph edges
Optimization variables	
$x_i^{jk} \in \{0,1\}$	1 if vehicle i travels arc (j,k), 0 otherwise
$Q_i^j \in \mathbb{R}_{\geq 0}$	Load of vehicle i when leaving vertex j
$B_i^j \in \mathbb{R}_{\geq 0}$	Beginning of service of vehicle i at vertex j
Problem data	
$c_i^{jk} \in \mathbb{R}_{\geq 0}$	Incurred cost if vehicle i travels arc (j,k)
$q^j \in \mathbb{R}$	Demand/supply at location $j \in R$
$C_i \in \mathbb{R}_{\geq 0}$	Capacity of vehicle i
$t_i^{jk} \in \mathbb{R}_{\geq 0}$	Travel time from j to k for vehicle i
$d^j \in \mathbb{R}_{\geq 0}$	Service duration at $j \in V_A$
$Q_i^{\text{init}} \geq 0$	Initial load of vehicle i

where $\underline{Q}^j = \max\{0, q^j\}$, $\overline{Q}_i^j = \min\{C_i, C_i + q^j\}$ and $Q_i^{\text{init}} \in \mathbb{R}_{\geq 0}$. We make the standing assumption that problem (1) is feasible and admits an optimal solution. Throughout the document, we use the convention that subscripts denote the vehicle index, while superscripts refer to locations. Table I collects all the relevant symbols. Notice that, in order to satisfy (1j), it is necessary to assume $C_i \geq \max_{j\in R}\{q^j\}$, i.e., the generic task can be performed by at least one robot. In order to keep the discussion not too technical, in the following we always maintain this assumption. However, note that the algorithm proposed in this paper works also if this assumption is removed. A discussion on this extension is given in Section V-A, from which it follows that trivial solutions where a robot is assigned all the tasks are not feasible.

We conclude by noting that problem (1) is mixed-integer but not linear. Indeed, the constraints (1h)–(1i) are nonlinear. However, an equivalent linear formulation of these constraints is always possible (the detailed procedure is outlined in Appendix A). In the rest of the paper, we refer to problem (1) as being a MILP, with the implicit assumption that the constraints (1h)–(1i) are replaced by their linear version. Notice that it is not necessary to perform such reformulation when implementing the algorithm. Indeed, several modern solvers allow for the implementation of these constraints by means of so-called *indicator constraints*. We however prefer to consider a linear reformulation in order to streamline the analysis.

B. Description of Cost and Constraints

Let us detail the cost and constraints of problem (1). The objective (1a) minimizes the total route cost, in particular, the total euclidean distance traveled by robots.

Constraint (1b) enforces that every location has to be visited at least once. Typically, PDVRP formulations consider this constraint as an equality, however, in the considered case of cost being the total distance, the solution is the same both with the equality and with the inequality. We stick to the inequality formulation as this allows us to exploit the problem

structure and efficiently solve the problem. Constraints (1c)–(1d) guarantee that every vehicle starts its mission at s and ends at σ. Equality (1e) is a flow conservation constraint, meaning that if a vehicle enters a location k it also has to leave it. Constraint (1f) ensures that, if a robot i performs a pickup operation, it also has to perform the corresponding delivery. Inequality (1g) imposes that deliveries have to occur after pickups. Constraint (1h) avoids subtours in each vehicle route (i.e. paths passing from the same location more than once), while inequalities (1i)–(1j) ensure that the total vehicle capacity is never exceeded. Finally, (1k) takes into account the initial load of the i-th robot.

III. Distributed Algorithm

In this section, we propose our distributed algorithm to solve the Multi-vehicle PDVRP. We first present the distributed problem setup. Then, we formally describe the proposed distributed algorithm.

A. Toward a Distributed Resource Allocation Scheme

We assume robots aim to solve problem (1) in a distributed fashion, i.e. without a central (coordinating) node. In order to solve the problem, we suppose each robot is equipped with its own communication and computation capabilities. Robots can exchange information according to a static communication network modeled as a connected and undirected graph $\mathcal{G} = (\{1, \dots, N\}, \mathcal{E})$. The graph $\mathcal{G}$ models the communication in the sense that there is an edge $(i, \ell) \in \mathcal{E}$ if and only if robot i is able to send information to robot ℓ. For each node i, the set of neighbors of i is denoted by $\mathcal{N}_i = \{\ell \in \mathbb{I} : (i, \ell) \in \mathcal{E}\}$. We consider the challenging scenario in which the i-th robot only knows problem data related to it, namely the i-th travelling times t_i^{jk}, the i-th local capacity C_i and the i-th cost entries c_i^{jk}, thus not having access to the entire problem formulation. However, we reasonably assume that all the robots know the demand/supply values q_j and service time d_j for each task request $j \in V_A$.

Note that the optimization variables in (1) associated with a robot i are all and only the variables with subscript i (i.e. x_i^{jk}, B_i^j and Q_i^j for all j, k). In order to simplify the notation, let us define for all i the set

$$Z_i = \Big\{(x_i, B_i, Q_i) \text{ such that (1c)–(1l) are satisfied}\Big\}. \quad (2)$$

Indeed, note that the constraints (1c)–(1l) are repeated for each index i. With this shorthand, any route that robot i can implement can be denoted more shortly as $(x_i, B_i, Q_i) \in Z_i$. However, note that feasible solutions to problem (1) must not only be valid routes, but they must also satisfy the pickup/delivery demand. More formally, the vectors (x_i, B_i, Q_i) must satisfy both the constraints

$$(x_i, B_i, Q_i) \in Z_i \qquad \forall i \in \mathbb{I}, \quad (3a)$$

$$\text{and} \quad \sum_{i=1}^{N} \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq 1 \qquad \forall j \in R, \quad (3b)$$

where (3b) is the coupling constraint (1b). In light of this observation, the main idea to devise a distributed algorithm for problem (1) is to perform a negotiation to determine the value of the left-hand side of the constraint (3b) in a distributed way. This technique is known as *primal decomposition* (see also Appendix B). Let us define *allocation* vectors $y_i \in \mathbb{R}^{|R|}$ that add up to the right-hand side of (3b), i.e.

$$\sum_{i=1}^{N} [y_i]_j = 1, \qquad \forall j \in R,$$

where $[y_i]_j$ denotes the j-th component of y_i. The variables y_i should be interpreted as the allocation of a resource which is shared among the robots (cf. Appendix B). Each robot i will aim to determine its allocation y_i in such a way that

$$\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq [y_i]_j, \qquad \forall j \in R,$$

from which it directly follows that (3b) is satisfied. As it will be clear from the forthcoming analysis, the j-th entry of the vector y_i determines whether or not robot i must perform task j. In the next subsection, we introduce our distributed algorithm, whose purpose is to coordinate the computation of allocation vectors y_i. such that (3b) is satisfied.

B. Distributed Algorithm Description

Let us now introduce our distributed algorithm. Let $t \in \mathbb{N}$ denote the iteration index. Each robot i maintains an estimate of the local allocation vector $y_i^t \in \mathbb{R}^{|R|}$. At each iteration, the vector y_i^t is updated according to (4)–(5). After a finite number of iterations, say $T_f \in \mathbb{N}$, the robots compute a tentative solution to the PDVRP based on the last computed allocation $y_i^{T_f}$ with (6)–(7). The whole algorithm can be seen as a subgradient method applied to a suitable, convex reformulation of problem (1). Algorithm 1 summarizes the scheme as performed by each robot i. The symbol $\text{conv}(Z_i)$ denotes the convex hull of the set Z_i, while α^t is a step-size sequence. The algorithm has also some tuning parameters that are reported on the top of the table.

Let us informally comment on the algorithm table. The algorithm is composed of two logic blocks. The first block, represented by the steps (4)–(5), is repeated in an iterative manner and is aimed at computing a final allocation vector $y_i^{T_f}$. Notice that in (4) we replace the mixed-integer set Z_i with a convex, polyhedral set. This makes the problem easier to solve. Moreover, thanks to the Shapley-Folkman lemma, part of the optimal decision variables for (4) satisfy the binary constraints. The integrality of the remaining optimization variables is recovered at the end of the algorithm in (7). This significantly reduces the computational burden of the scheme. As for the update in (5), it is a subgradient-based iteration on the resource allocation variable y_i. An analysis of this scheme is provided in Section IV-B. In the second block, represented by the steps (6)–(7), the final allocation is used to determine a solution to the original problem. The thresholding step (6) rectifies the current local allocation $y_i^{T_f}$ to obtain the final allocation y_i^{END}, which is fed to problem (7) to compute the local portion of solution $(x_i^{\text{END}}, B_i^{\text{END}}, Q_i^{\text{END}})$. An analysis of the algorithm is provided in Section IV, while a discussion

Algorithm 1 Distributed Resource Allocation for PDVRP

Parameters: $T_f > 0$, $M > 0$, $0 < \delta < 1$.

State: $y_i \in \mathbb{R}^{|R|}$ (initialized at $[y_i^0]_j = \delta/N$ for all $j \in R$).

Repeat for $t = 0, 1, \dots, T_f - 1$:

Compute μ_i^t as Lagrange multiplier of linear program

$$\begin{aligned} \min_{x_i, B_i, Q_i, v_i} \quad & \sum_{(j,k)\in\mathcal{E}_A} c_i^{jk} x_i^{jk} + M v_i \\ \text{subj. to} \quad & \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq [y_i^t]_j - v_i, \qquad \forall j \in R \\ & (x_i, B_i, Q_i) \in \text{conv}(Z_i), \;\; v_i \geq 0 \end{aligned} \tag{4}$$

Receive μ_ℓ^t from neighbors $\ell \in \mathcal{N}_i$ and update

$$y_i^{t+1} = y_i^t - \alpha^t \sum_{\ell\in\mathcal{N}_i} \left(\mu_i^t - \mu_\ell^t\right) \tag{5}$$

Perform component-wise thresholding of allocation

$$[y_i^{\text{END}}]_j = \min\left([y_i^{T_f}]_j, 1\right), \qquad \forall j \in R \tag{6}$$

Return $(x_i^{\text{END}}, B_i^{\text{END}}, Q_i^{\text{END}})$ as optimal solution of MILP

$$\begin{aligned} \min_{x_i, B_i, Q_i} \quad & \sum_{(j,k)\in\mathcal{E}_A} c_i^{jk} x_i^{jk} \\ \text{subj. to} \quad & \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq [y_i^{\text{END}}]_j, \qquad \forall j \in R \\ & (x_i, B_i, Q_i) \in Z_i \end{aligned} \tag{7}$$

on the choice of the tunable parameters M, T_f and δ can be found in Section V-A.

A few additional remarks are in order. First, note that the only information exchanged with neighbors are the Lagrange multipliers μ_i^t. Thus, robots never exchange sensitive local information (such as the cost, the load or the computed routes), therefore the algorithm maintains privacy. Remarkably, even though the number of robots increase, the amount of local computation remains unchanged. This scalability property is attractive and enables the solution of the problem even in large networks of robots.

In order to state the theoretical properties of the algorithm, we make the following standard assumption on the step-size α^t appearing in (5).

Assumption III.1. *The step-size sequence $\{\alpha^t\}_{t\geq 0}$, with each $\alpha^t \geq 0$, satisfies $\sum_{t=0}^{\infty} \alpha^t = \infty$, $\sum_{t=0}^{\infty} \left(\alpha^t\right)^2 < \infty$.* □

A discussion on possible choices of the step-size satisfying Assumption III.1 is given in Section V-A. The next theorem represents the central result of the paper.

Theorem III.2 (Finite-time feasibility)**.** *Let Assumption III.1 hold and let $0 < \delta < 1$. Then, for a sufficiently large $M > 0$, there exists a time $T_\delta > 0$ such that the vector $(z_1^{\text{END}}, \dots, z_N^{\text{END}})$, the aggregate output of Algorithm 1, with each $z_i^{\text{END}} = (x_i^{\text{END}}, B_i^{\text{END}}, Q_i^{\text{END}})$, is a feasible solution to problem* (1)*, provided that the total iteration count satisfies $T_f \geq T_\delta$.* □

The proof is provided in Section IV-D. In Section VI, we perform an empirical study on the optimality of the solutions computed by Algorithm 1.

Remark III.3 (Algorithm complexity)**.** *Note that the steps* (6)–(7) *are performed only once at the end of the algorithm. Steps* (4)–(5) *instead are performed T_f times. Let L_i denote the number of constraints of* $\text{conv}(Z_i)$*. Then, the average complexity of* (4) *(solved via the simplex algorithm) is $\mathcal{O}(L_i + |R|)$. Step* (5) *consists of sums and multiplications and its complexity is $\mathcal{O}(|\mathcal{N}_i||R|)$. Step* (6) *is a thresholding with complexity $\mathcal{O}(|R|)$. Excluding the MILP* (7)*, the total algorithm complexity is thus $\mathcal{O}(T_f(|\mathcal{N}_i||R| + L_i))$. Finally,* (7) *can be solved using a branch-and-bound scheme, which complexity is exponential in the number $O(|R|^2)$ of integer variables.* □

IV. Algorithm Analysis

In this section, we provide a theoretical study of Algorithm 1. The algorithm inherits the ideas of [30], however here we are considering an enlargement of the constraints rather than a restriction, and moreover there is a thresholding operation which is specific of Algorithm 1. We provide a compact analysis focused on those properties that are peculiar to the PDVRP and that are not considered elsewhere. To arrive at the final result given by Theorem III.2, we will proceed with the following steps.

(i) We show that the steps in (4) and (7) are well posed (Section IV-A).
(ii) We analyze the steps in (4) and (5), needed to retrieve an optimal solution to a relaxed PDVRP (Section IV-B).
(iii) Finally, we prove Theorem III.2 (Section IV-D) with the help of auxiliary technical lemmas (Section IV-C).

From now on, we denote by $\mathbf{1}$ the vector of ones with appropriate dimension.

A. Feasibility of Local Problems

We begin the analysis by proving that the algorithm is well posed. In particular, we show that it is indeed possible to solve the problems (4) and (7). The next lemma formalizes feasibility of problem (4).

Lemma IV.1. *Consider a robot $i \in \mathbb{I}$ and let y_i^t be allocation computed by Algorithm 1 at an iteration t. Then, problem* (4) *is feasible.*

Proof. See Appendix C-A. □

Proving feasibility of problem (7) is more delicate and relies upon the thresholding operation, as formally shown next.

Lemma IV.2. *Consider a robot $i \in \mathbb{I}$ and let y_i^{END} be the final allocation computed by Algorithm 1. Then, problem* (7) *is feasible.*

Proof. See Appendix C-B. □

B. Convergence of Distributed Resource Allocation Scheme

We now focus on the first logic block of Algorithm 1, namely steps (4)–(5). These two iterative steps can be used to obtain an optimal allocation associated with the linear program

$$\begin{aligned} \min_{x,B,Q} \; & \sum_{i=1}^{N} \sum_{(j,k)\in\mathcal{E}_A} c_i^{jk} x_i^{jk} \\ \text{subj. to} \; & \sum_{i=1}^{N} \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq \delta \qquad \forall j \in R \\ & (x_i, B_i, Q_i) \in \text{conv}(Z_i), \qquad \forall\, i \in \mathbb{I}. \end{aligned} \tag{8}$$

Formally, we denote with $(y_1^{\text{CONV}}, \ldots, y_N^{\text{CONV}})$ such optimal allocation, which is the optimal vector satisfying the conditions specified in Section III-A. Before stating formally this result, we note that, by the discussion in Section III-A, problem (8) is essentially the same as problem (1), except that the mixed-integer constraints $(x_i, B_i, Q_i) \in Z_i$ are replaced by their convex relaxation $(x_i, B_i, Q_i) \in \text{conv}(Z_i)$. Moreover, the coupling constraints are enlarged by changing the right-hand side from 1 to $\delta \in (0,1)$ (recall that δ is a tunable parameter of Algorithm 1).

Now, denote by $(z_1^{\text{CONV}}, \ldots, z_N^{\text{CONV}})$ an optimal solution of problem (8) with each $z_i = (x_i, B_i, Q_i)$, and by $\{y_1^t, \ldots, y_N^t\}_{t\geq 0}$ the allocation vector sequence produced by (4)–(5). The following lemma summarizes the convergence properties of the distributed resource allocation scheme (4)–(5).

Lemma IV.3. *Let Assumption III.1 hold and recall that $y_i^0 = \delta/N\mathbf{1}$ for all $i \in \mathbb{I}$. Moreover, let $(y_1^{\text{CONV}}, \ldots, y_N^{\text{CONV}}) \in \mathbb{R}^{N|R|}$ be an optimal allocation associated with problem* (8)*, i.e., a vector satisfying $\sum_{i=1}^N y_i^{\text{CONV}} = \delta\mathbf{1}$ and $\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq [y_i^{\text{CONV}}]_j$ for all $j \in R$ and $i \in \mathbb{I}$. Then, for a sufficiently large $M > 0$, the distributed algorithm* (4)–(5) *generates a sequence $\{y_1^t, \ldots, y_N^t\}_{t\geq 0}$ such that*

(i) $\sum_{i=1}^N y_i^t = \delta\mathbf{1}$*, for all $t \geq 0$;*
(ii) $\lim_{t\to\infty} \|y_i^t - y_i^{\text{CONV}}\| = 0$ *for all $i \in \mathbb{I}$.*

Proof. We refer the reader to [30] for the proof. □

C. Intermediate Results

Before turning to the proof of Theorem III.2, we provide some preparatory lemmas. The first lemma justifies the use of δ (an arbitrarily small positive number) in place of the original right-hand side 1 in the coupling constraints (1b).

Lemma IV.4. *For all $i \in \mathbb{I}$, let $(x_i, B_i, Q_i) \in Z_i$ such that $\sum_{i=1}^N \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} > 0$ for all $j \in R$. Then, it holds $\sum_{i=1}^N \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq 1$ for all $j \in R$.*

Proof. For each component $j \in R$, by assumption we have

$$\sum_{i=1}^{N} \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} > 0.$$

Note that, since each $x_i^{jk} \in \{0,1\}$, the quantity $\sum_{i=1}^N \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk}$ is either zero or at least equal to 1. Therefore, because of the assumption, we have $\sum_{i=1}^N \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq 1$ for all $j \in R$. □

The next two lemmas will be used in the sequel to characterize an optimal allocation associated with problem (8).

Lemma IV.5. *For all $i \in \mathbb{I}$, let $\tilde{y}_i \in \mathbb{R}^{|R|}$ and $(\tilde{x}_i, \tilde{B}_i, \tilde{Q}_i) \in \text{conv}(Z_i)$ such that $\sum_{k:(j,k)\in\mathcal{E}_A} \tilde{x}_i^{jk} > [\tilde{y}_i]_j$ for all $j \in R$. Then, there exists $(\bar{x}_i, \bar{B}_i, \bar{Q}_i) \in Z_i$ satisfying $\sum_{k:(j,k)\in\mathcal{E}_A} \bar{x}_i^{jk} > [\tilde{y}_i]_j$ for all $j \in R$.*

Proof. Fix a robot $i \in \mathbb{I}$ and note that, because of the flow constraints (1e) and the subtour elimination constraints (1h), for all $j \in R$ it holds

$$\max_{(x_i,B_i,Q_i)\in Z_i} \left(\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \right) = 1. \tag{9}$$

Moreover, note that it is possible to choose a local solution passing through all the locations (possibly at a high cost), i.e., there exists $(\bar{x}_i, \bar{B}_i, \bar{Q}_i) \in Z_i$ such that $\sum_{k:(j,k)\in\mathcal{E}_A} \bar{x}_i^{jk} = 1$ for all $j \in R$. Therefore, for all $j \in R$ it holds

$$\begin{aligned} & \sum_{k:(j,k)\in\mathcal{E}_A} \bar{x}_i^{jk} \\ & = \max_{(x_i,B_i,Q_i)\in Z_i} \left(\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \right) \\ & \overset{(a)}{=} \max_{(x_i,B_i,Q_i)\in \text{conv}(Z_i)} \left(\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \right) \\ & \geq \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \qquad \text{for all } (x_i, B_i, Q_i) \in \text{conv}(Z_i), \end{aligned} \tag{10}$$

where *(a)* follows by linearity of the cost. In particular, the previous inequality holds with $(x_i, B_i, Q_i) = (\tilde{x}_i, \tilde{B}_i, \tilde{Q}_i)$ and thus for all $j \in R$ we have $\sum_{k:(j,k)\in\mathcal{E}_A} \bar{x}_i^{jk} \geq \sum_{k:(j,k)\in\mathcal{E}_A} \tilde{x}_i^{jk} > [\tilde{y}_i]_j$. □

Lemma IV.6. *Let $(y_1^{\text{CONV}}, \ldots, y_N^{\text{CONV}}) \in \mathbb{R}^{N|R|}$ be an optimal allocation associated with problem* (8)*, i.e., a vector satisfying $\sum_{i=1}^N y_i^{\text{CONV}} = \delta\mathbf{1}$ and $\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq [y_i^{\text{CONV}}]_j$ for all $j \in R$ and $i \in \mathbb{I}$. Then, $y_i^{\text{CONV}} \leq \mathbf{1}$ for all $i \in \mathbb{I}$.*

Proof. By contradiction, suppose that there is a component $j \in R$ for which $[y_i^{\text{CONV}}]_j > 1$. By assumption, we have $\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq [y_i^{\text{CONV}}]_j$. Using Lemma IV.5, we conclude that there exists $(\bar{x}, \bar{B}, \bar{Q}) \in Z_i$ such that

$$\sum_{k:(j,k)\in\mathcal{E}_A} \bar{x}_i^{jk} \geq [y_i^{\text{CONV}}]_j > 1,$$

which contradicts (9). □

D. Proof of Theorem III.2

First, note that, by Lemmas IV.1 and IV.2, the algorithm is well posed. Moreover, by construction (cf. problem (7)) it holds $(x_i^{\text{END}}, B_i^{\text{END}}, Q_i^{\text{END}}) \in Z_i$ for all $i \in \mathbb{I}$. Therefore, all the local constraints (1c) to (1l) are satisfied by $(x_i^{\text{END}}, B_i^{\text{END}}, Q_i^{\text{END}})$ and we only need to show that there exists $T_\delta > 0$ such that constraint (1b) is satisfied by

$(x_i^{\text{END}}, B_i^{\text{END}}, Q_i^{\text{END}})$ if $T_f \geq T_\delta$. By Lemma IV.4, it suffices to prove that $\sum_{i=1}^N \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} > 0$ for all $j \in R$.

Consider the auxiliary sequence $\{y_1^t, \dots, y_N^t\}_{t\geq 0}$ generated by Algorithm 1. By Lemma IV.3, this sequence converges to the vector $(y_1^{\text{CONV}}, \dots, y_N^{\text{CONV}})$. By definition of limit (using the infinity norm), there exists $T_\delta > 0$ such that $\|y_i^t - y_i^{\text{CONV}}\|_\infty < \delta/N$ (and thus $y_i^t < y_i^{\text{CONV}} + \delta/N\mathbf{1}$) for all $i \in \mathbb{I}$ and $t \geq T_\delta$.

Let us define a vector $\rho_i \in \mathbb{R}^{|R|}$ representing the mismatch between $y_i^{T_f}$ and its thresholded version y_i^{END},

$$\rho_i = y_i^{T_f} - y_i^{\text{END}}, \qquad \text{for all } i \in \mathbb{I}. \tag{11}$$

By definition (6), it holds $\rho_i \geq 0$. Then, for all $j \in R$, it holds

$$\begin{aligned}\sum_{i=1}^N \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{\text{END}jk} &\geq \sum_{i=1}^N [y_i^{\text{END}}]_j \\ &= \underbrace{\sum_{i=1}^N [y_i^{T_f}]_j}_{\delta} - \sum_{i=1}^N [\rho_i]_j \\ &= \delta - \sum_{i=1}^N [\rho_i]_j\end{aligned}$$

Let us temporarily assume that $\rho_i < \delta/N\mathbf{1}$ for all $i \in \mathbb{I}$. Then, we obtain the desired statement

$$\begin{aligned}\sum_{i=1}^N \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{\text{END}jk} &\geq \delta - \sum_{i=1}^N [\rho_i]_j \\ &> \left(\delta - \sum_{i=1}^N \delta/N\right) = 0.\end{aligned}$$

It remains to show that $\rho_i < \delta/N\mathbf{1}$ for all $i \in \mathbb{I}$. Fix a robot i and consider a component $j \in R$ of the vector ρ_i. Owing to the definition (6) of y_i^{END}, either the j-th component is equal to $[y_i^{T_f}]_j$ or it is equal to 1. In the former case, we have $[\rho_i]_j = 0 < \delta/N$. In the latter case, there is a non-negative mismatch $[\rho_i]_j = [y_i^{T_f}]_j - 1 \geq 0$. Now, using the fact $y_i^t < y_i^{\text{CONV}} + \delta/N\mathbf{1}$ for all $t \geq T_\delta$, we have

$$\begin{aligned}[\rho_i]_j &= [y_i^{T_f}]_j - 1 \\ &< [y_i^{\text{CONV}}]_j - 1 + \delta/N \\ &\leq \delta/N\end{aligned}$$

provided that $T_f \geq T_\delta$, where in the last inequality we applied Lemma IV.6. The proof follows. □

V. Discussion and Extension

In this section, we provide guidelines for the choice of the algorithm parameters and we discuss a possible extension of Algorithm 1 to a more general setting.

A. On the Choice of the Parameters

As already mentioned in Section III, there are a few parameters that must be appropriately set in order for Algorithm 1 to work correctly. The basic requirements for the parameters are summarized in Theorem III.2 and are recalled here: *(i)* $M > 0$ must be sufficiently large, *(ii)* δ is any number in the open interval $(0, 1)$, *(iii)* the total number of iterations $T_f > 0$ must be sufficiently large, *(iv)* the step-size sequence $\{\alpha^t\}_{t\geq 0}$ must satisfy Assumption III.1.

The parameter $M > 0$ is an exact penalty weight (cf. [31]). The minimum admissible value is the 1-norm of the dual solution of problem (8). A conservative choice is any large number not creating numerical instability when solving the local problems (4). The assumption on the step-size sequence $\{\alpha^t\}_{t\geq 0}$ is typical in the distributed optimization literature. A sensible choice satisfying Assumption III.1 is $\alpha^t = K/(t+1)$, with any $K > 0$.

The purpose of the parameter δ is to enlarge the constraint (1b) (see also Lemma IV.4) and is linked to the minimum value of T_f, i.e., the minimum number of iterations to guarantee feasibility (cf. Theorem III.2). There is an inherent tradeoff between δ and the minimal T_f. For δ close to 1, precedence is given to feasibility and T_f may become smaller, while for δ close to 0, solution optimality is prioritized, possibly at the cost of a higher T_f. Indeed, for δ close to 1, the time T_δ in the proof of Theorem III.2 may be smaller, and therefore a smaller number of iterations T_f may be sufficient to attain feasibility of the solution. Instead, for δ close to 0, a greater number of iterations may be required to obtain feasibility. However, in the latter case there could be less robots for which the components of $y_i^{T_f}$ are positive (because, by Lemma IV.3, it must hold $\sum_{i=1}^N y_i^{T_f} = \delta\mathbf{1}$ with a small positive δ), thus forcing less robots to pass through the same location. In Section VI, we perform simulations in order to study the trade-off between δ and T_f numerically.

B. Extension to Heterogeneous PDVRP Graphs

Let us outline a possible extension of problem (1) that can be handled by Algorithm 1. Recall from Section II-B that problem (1) has the implicit assumption that $C_i \geq \max_{j\in R}\{q^j\}$ for all $i \in \mathbb{I}$, where C_i is the capacity of vehicle i and q^j is the demand/supply at location j. In real scenarios, while there may be vehicles potentially capable of performing all the pickup/delivery requests, it is often the case that many vehicles are small sized and can only accomplish a subset of the task requests. This means that the assumption $C_i \geq \max_{j\in R}\{q^j\}$ may not hold for some robots.

The general case just outlined can be handled with minor modifications in the formulation of problem (1) and in the algorithm. Indeed, if $C_i < q^j$ for some robot i and some location j, the PDVRP (1) would be unsolvable by construction (due to infeasibility). Thus, for each robot $i \in \mathbb{I}$ we define the largest *local* set of requests $R_i \subseteq R$ such that $C_i \geq q^j$ for all $j \in R_i$. Following the description in Section II-A, the sets R_i will now induce *local* graphs $\mathcal{G}_{Ai} = (V_{Ai}, \mathcal{E}_{Ai})$ of possible paths, with vertex set $V_{Ai} = \{s, \sigma\} \cup R_i$ and edge set $\mathcal{E}_{Ai} = \{(j,k) \mid j,k \in V_{Ai}, j \neq k \text{ and } j \neq \sigma, k \neq s\}$. Then, each robot i defines a smaller set of optimization variables x_i^{jk} with $j,k \in V_{Ai}$ (instead of $j,k \in V_A$) and similarly for B_i and Q_i. The optimization problem is formulated similarly to problem (1) by dropping all the references to non-existing optimization variables. The resulting modified version of problem (1) is now feasible as long as for all $j \in R$ there exists

$i \in \mathbb{I}$ such that $C_i \geq q^j$ (i.e., each task can be performed by at least one robot).

The distributed algorithm also requires minor modifications. In particular, the summations in problems (4) and (7) are performed using $\mathcal{E}_{Ai}$ in place of $\mathcal{E}_A$. Moreover, the thresholding operation (6) is replaced by the following one

$$[y_i^{\text{END}}]_j = \begin{cases} \min\left([y_i^{T_f}]_j, 1\right) & \text{if } j \in R_i \\ \min\left([y_i^{T_f}]_j, 0\right) & \text{otherwise} \end{cases}$$

for all $j \in R$. Finally, the sets Z_i must be replaced by the new version of the constraints (1c)–(1l) with $\mathcal{E}_{Ai}$ in place of $\mathcal{E}_A$. It is possible to follow essentially the same line of proof outlined in Section IV with only minor precautions in Lemmas IV.2 and IV.5, by which it can be concluded that Theorem III.2 holds with no changes.

VI. Simulations on Gazebo

In this section, we provide simulation results for the proposed distributed algorithm for teams of TurtleBot3 Burger ground robots that have to serve a set of pickup and delivery requests scattered in the environment. All the simulations are performed using the ChoiRbot [32] ROS 2 framework. We first describe how we integrate the proposed distributed scheme in ChoiRbot and then we show simulation results.

A. Simulation Set-up

In the ChoiRbot software architecture, each robot is modeled as a cyber-physical agent and consists of three interacting layers: distributed optimization layer, trajectory planning layer and low-level control layer. The ChoiRbot architecture is based on the novel ROS 2 framework, which handles interprocess communications via the TCP/IP stack. This allows us to implement the proposed distributed scheme on a real WiFi network in which robots communicate with few neighbors according to a given graph. Robots are simulated in the Gazebo environment [33], which provides an accurate estimation of the TurtleBot dynamics. The resulting simulations are so accurate that the experiments performed in the next section are obtained without any change or tuning in the code. In this sense, the proposed results are comparable to experimental results on a real team of robots. In Figure 2, we show a snapshot of one of the simulations addressed in the next section. Due to packet losses in the ROS 2 communication middleware with a large number of nodes on a single machine, we were not able to run simulations with more than 30 robots.

We now describe more in detail the components of the proposed architecture. As said, the distributed optimization layer handles the cooperative solution of the pickup and delivery problem. It consists of a set of optimization processes, one for each robot, that perform the steps of Algorithm 1. At the beginning of the simulation, each optimization node gathers the information on the pickup and delivery requests and evaluates the cost vector c_i (i.e. the robot-to-task distances) and the local constraint sets Z_i. We stress that these computations are performed independently for each robot on different processes, without having access to the other robot information. After the initialization, robots start communicating and performing the

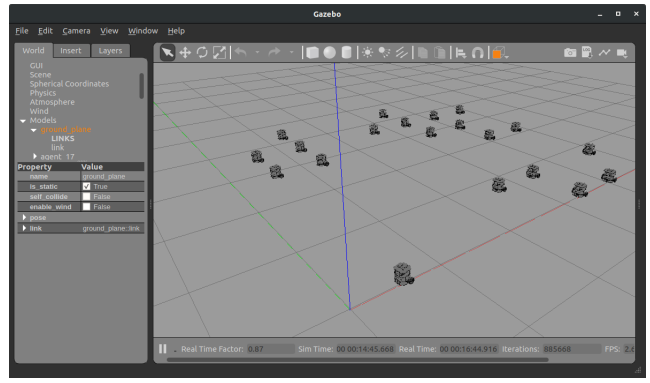

Figure 2. Snapshot of the initial condition of one of the Gazebo simulations.

steps of the distributed algorithm proposed in Section III-B. To implement Algorithm 1, we used the DISROPT Python package [34], which provides the needed features to encode the algorithm steps and is compatible with the ChoiRbot framework (see also [32]). As soon as the distributed optimization procedure completes, robots start moving towards the assigned tasks. Due to constraint (1b), suboptimal solutions of problem (1) may lead more than one robot to perform the same task. Thus, the robot communicates to a so-called Auth node that it wants to start a particular task. If the task has been already taken care of by another robot, the Auth node denies authorization and the robot performs the next one. In such a way, redundant assignments are avoided. The target positions are then communicated to the local trajectory planning layer and then fed to the low-level controllers to steer the robots over the requests positions. The controller nodes interact with Gazebo, which simulates the robot dynamics and provides the pose of each robot.

We have experimentally found that a satisfying tuning of the distributed algorithm is as follows. The robots perform 250 iterations with local allocation initialized as in Algorithm 1. For the first 125 iterations they use the diminishing step size $\alpha^t = 0.005/(t+1)$, then they use a constant step size (equal to the last computed one). Because of the constant step size, the final allocation fed to the thresholding operation (6) is the running average computed from iteration 126 on, i.e.

$$\left(\sum_{\tau=126}^{250} \alpha^\tau y_i^\tau\right) \Big/ \left(\sum_{\tau=126}^{250} \alpha^\tau\right)$$

This particular tweaking allows the robots to quickly converge to a good-quality solution.

B. Results

We performed 3 Monte Carlo simulations on random instances of problem (1) on the described platform with TurtleBot3 robots. We test the behavior of the proposed scheme when both the number of requests $|R|$ and the number of robots N are varied. In this way it is possible to assess the performance of the algorithm if $|R| > N$ or if $|R| < N$.

First simulation. To begin with, we test optimality of the solution computed by the algorithm while varying the number of robots N. We perform 50 Monte Carlo trials for each value

of N and we fix $\delta = 0.1$ to prioritize optimality over feasibility (cf. Section V-A). For each trial, 10 pickup requests and 10 corresponding deliveries are randomly generated on the plane. In Figure 3, we show the cost error of the solution actuated by robots after 250 iterations of the distributed algorithm (in blue), compared to the cost of a centralized solver, with varying number of robots. The distributed algorithm achieves an average $30 - 40\%$ suboptimality.

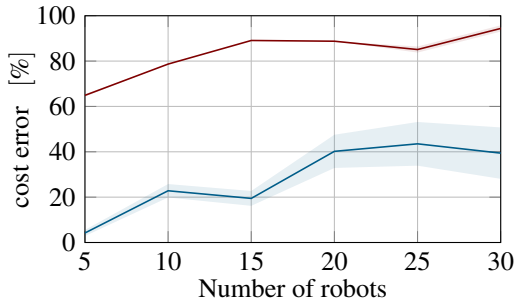

Figure 3. Cost error in Monte Carlo simulations on Gazebo for varying number of robots. Blue: proposed approach. Red: baseline approach. The shaded areas represent one standard deviation.

Comparison with distributed baseline approach. The solutions are confronted with a distributed greedy market-based algorithm as follows. Upon receiving the task requests and filtering out those that cannot be performed due to insufficient load capacity, each robot initially self-assigns the pickup task closest to its position together with the corresponding delivery. Then, it self-assigns the next pickup task closest to the last delivery location, and so on until the task list is empty. To remove ties, robots compute the cost of performing each pickup-delivery pair, and then perform a min-consensus algorithm to decide the robot with the lower cost. This algorithm is run up to its convergence to the best attainable value. The cost error obtained with this approach is depicted in Figure 3 (in red). From the figure, it emerges that our approach outperforms the baseline since it achieves lower suboptimality levels.

Second simulation. Now, we assess the behavior of the cost error while varying the total number of requests. Specifically, we consider a team of 20 robots and we let the number of requests $|R|$ vary from 4 to 24 (with $\delta = 0.9$). For each of these values of $|R|$, we perform 50 trials and we let the robots implement the solution after 250 iterations of the distributed algorithm. The results are depicted in Figure 4 and Figure 5. Notably, as it can be seen from Figure 4, the mean relative error remains constant while increasing the number of requests. This is an appealing feature of the proposed strategy considering the fact that, as depicted also in Figure 5, the global optimal cost increases with the number of tasks.

Third simulation. Finally, we perform simulations to determine the number of iterations needed to achieve finite-time feasibility while varying the number of robots N and the value of δ. We employed three values of δ, namely 0.1, 0.5, 0.9, and performed 50 trials for each of the values $N = 5, 10, 15, 20, 25, 30$ and for each value of δ. In each trial, we performed 250 iterations of the algorithm and recorded the value of the coupling constraint. Then we determined the

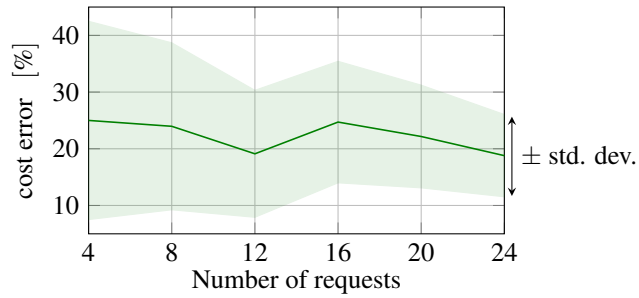

Figure 4. Cost error in Monte Carlo simulations on Gazebo for varying number of requests.

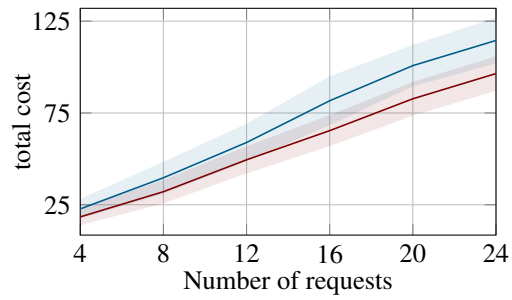

Figure 5. Comparison between the centralized optimal solution (red) and the one found by the proposed distributed strategy (blue).

following quantity

$$\min\ t$$
$$\text{such that} \sum_{i=1}^{N} \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk^\tau} \geq 1 \qquad \text{for all } \tau \geq t,$$

which is essentially an empirical value of T_δ appearing in Theorem III.2. Interestingly, we found out that, for all the trials, the empirical value of T_δ is zero, which means that in all the simulated scenarios the algorithm provides a feasible solution to the PDVRP (1) since the first iteration.

VII. Experiments

To conclude, we show experimental results on real teams of robots solving PDVRP instances. We first performed a small benchmark experiment to assess the performance of the algorithm and then we perform more complex experiments to showcase how it can be implemented on large fleets of robots.

A. Benchmark Experiment

In this experiment, we consider a fleet of 4 TurtleBot3 burger ground robots that have to serve 4 pickup tasks and their corresponding 4 deliveries. In this set-up, robots navigate in a cluttered environment containing obstacles. Robots start from initial positions on a line. Pickup and delivery tasks are generated on two different lines, so as to clearly distinguish the optimal robot-to-task assignment. To simulate the pickup/delivery procedure, each robot waits over the request location a random service time d^j between 3 and 5 seconds. The capacity C_i of each robot and the demand/supply q^j of tasks are drawn from uniform distributions. The velocity

of robots is approximately $0.2\,\mathrm{m/s}$. As regards the low-level controllers, we use a linear state feedback for single integrators. In this way, we can handle collision among robots and with obstacles via barrier functions using the approach described in [35]. Then, in order to get the unicycle inputs, we utilize a near-identity diffeomorphism (see [35]).

In Figure 6, we show a snapshot of the experiment set-up. We run the distributed algorithm for 1000 iterations and record the robot-to-task assignment and the cost along the algorithm evolution. Figure 7 reports the experiment analysis, while in Table II, we include a comparison of the allocation found by the proposed distributed algorithm and the one found via a centralized solver. The figure highlights that, as the algorithm evolves, the cost of the overall robot-to-task assignment decreases, and after a certain number of iterations the computed solution becomes conflict free (i.e., only one robot is assigned to each task). As it can be seen from the table, the solution computed by the distributed algorithm coincides with the optimal (centralized) solution. A video is also available as supplementary material to the paper.[1]

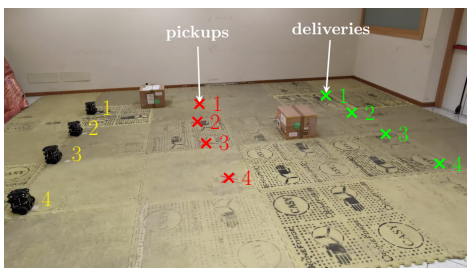

Figure 6. Benchmark experiment for the PDVRP problem. Robots start on a line. Pickups locations are denoted with red crosses, while deliveries are denoted with green crosses.

Table II
EXPERIMENT ANALYSIS: FINAL ASSIGNMENTS

Robot id	Distributed solution	Optimal solution
1	[1]	[1]
2	[2]	[2]
3	[3]	[3]
4	[4]	[4]

B. Large Experiments

We consider heterogeneous teams composed by Crazyflie nano-quadrotors and TurtleBot3 Burger mobile robots. Tasks are generated randomly in the space. In particular, we split the experiment area in two halves. Pickup requests are located in the right half, while deliveries are in the left half. Each robot can serve a subset of the pickup/delivery requests. This is decided randomly at the beginning of the experiment. The velocity of robots (both ground and aerial) is approximately $0.2\,\mathrm{m/s}$. The solution mechanism is the same used in the simulations.

As regards the low-level controllers of nano-quadrotors, a hierarchical controller has been considered. Specifically, a flatness-based position controller generates desired angular rates that are then actuated with a low-level PID control loop. The position controller receives as input a sufficiently smooth position trajectory, which is computed as a polynomial spline.

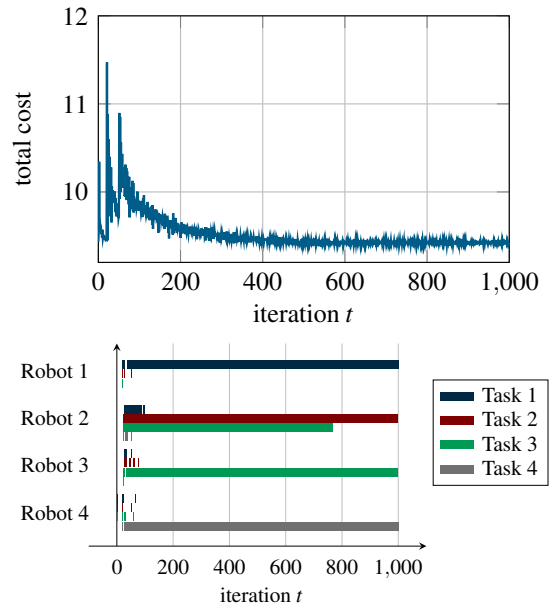

Figure 7. Analysis of benchmark experiment. Top: value of the cost along the algorithmic evolution, computed from problem (4). Bottom: assignment of robot to tasks along the algorithmic evolution.

We performed two different experiments. In the first one, there are 3 ground robots and 2 aerial robots that must serve a total of 5 pickups and 5 deliveries. In Figure 8, we show a snapshot from the experiment. Then we performed a second, larger experiment with 7 ground robots and 2 aerial robots that must serve 10 pickups and 10 deliveries. In Figure 9, we show snapshots from the second experiment. A video is also available as supplementary material to the paper.[2]

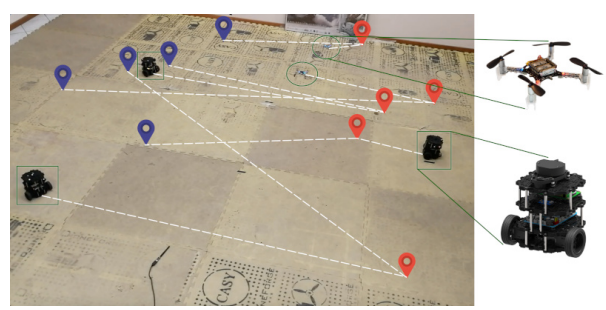

Figure 8. First experiment with ground and aerial robots for the PDVRP problem. Ground robots are indicated with a square, while aerial robots are delimited with circles. The red pins represent pickups, while the blue ones represent deliveries. The paths travelled by robots are depicted as dashed lines.

VIII. CONCLUSIONS

In this paper, we introduced a purely distributed scheme to address large-scale instances of the Pickup-and-Delivery Vehicle Routing Problem in networks of cooperating robots.

[1]The video can be also found at `https://youtu.be/tdtGNftdbng`.

[2]The video can be also found at `https://youtu.be/NwqzIEBNIS4`.

Figure 9. Snapshots from the second experiment. Left: robots have reached the pickup positions and perform the loading operation (simulated). Right: robots have reached the delivery positions and have terminated the mission.

The proposed distributed algorithm is shown to provide a feasible solution in a finite number of communication rounds and has remarkable scalability and privacy-preserving properties that allow for large robotic networks. The theoretical results are corroborated on a set of realistic simulations through a combined ROS 2 / Gazebo platform. Finally, experimental results on a real testbed highlight feasibility of the proposed solution for a heterogeneous team of ground and aerial robots. As a future line of research, a more complex set-up could be investigated. Indeed, variables as travel cost, service duration, travel time could be considered to be stochastic in order to obtain a more realistic model. Moreover, the approach can be enhanced to force assignments of single robots to each task and a convergence rate analysis can be performed.

Appendix A
Conversion to Mixed-Integer Linear Program

Problem (1) is almost a mixed-integer linear program, except for the fact that the constraints (1h) and (1i) are nonlinear. However, from a computational point of view, the constraints (1h) and (1i) can be readily recast as linear ones. To achieve this, we use a standard procedure (see [36], [37]) that can be summarized as follows.

First, we introduce for each i, j, k the constraints $0 \leq B_i^j \leq \overline{B}_i$, where $\overline{B}_i \geq 0$ is any conservative upper bound on the total travel time of vehicle i selected so as to preserve the solutions of the original problem [3]. While this operation does not affect the problem, it introduces a bound on the value of each B^j. After defining for all i, j, k the scalars $M_i^{jk} = \overline{B}_i + d^j + t_i^{jk}$ (or any larger number), the nonlinear constraint (1h) can be replaced with the linear one

$$B_i^k \geq B_i^j + d_j + t_i^{jk} - M_i^{jk}(1 - x_i^{jk}). \tag{12}$$

Equivalence of the constraint (1h) with (12) can be verified by noting that, for $x_i^{jk} = 1$, we obtain the desired constraint $B_i^k \geq B_i^j + d_j + t_i^{jk}$, while for $x_i^{jk} = 0$ the constraint (12) becomes $B_i^k \geq B_i^j + d_j + t_i^{jk} - M_i^{jk}$ (which is already implied by the constraints $B_i^k \geq 0$ and $B_i^j \geq 0$).

A similar reasoning can be applied to turn the constraint (1i) into a linear one. Let us define $\overline{W}_i^{jk} = \overline{Q}^j + q^k$ and $\underline{W}_i^{jk} = \overline{Q}^k - q^k - \underline{Q}^j$ (or any larger number), then constraint (1i) can be equivalently replaced with the pair of linear constraints

$$Q_i^k \geq Q_i^j + q^k - \overline{W}_i^{jk}(1 - x_i^{jk}), \tag{13a}$$
$$Q_i^k \leq Q_i^j + q^k + \underline{W}_i^{jk}(1 - x_i^{jk}). \tag{13b}$$

After introducing the additional constraints $0 \leq B_i^j \leq \overline{B}_i$ for all i, j, k and replacing (1h)–(1i) with their equivalent versions (12)–(13), problem (1) becomes a MILP.

[3] A simple possibility is to select $\overline{B}_i$ as the sum of all possible travel times t_i^{jk} from all i to all j, plus the service times d^j for all j.

Appendix B
Primal Decomposition

Consider a network of N agents indexed by $\mathbb{I} = \{1, \ldots, N\}$ that aim to solve a linear program of the form

$$\begin{aligned} \min_{x_1,\ldots,x_N} \quad & \sum_{i=1}^{N} c_i^\top x_i \\ \text{subj. to} \quad & x_i \in X_i, \qquad \forall i \in \mathbb{I}, \\ & \sum_{i=1}^{N} A_i x_i \leq b, \end{aligned} \tag{14}$$

where each $x_i \in \mathbb{R}^{n_i}$ is the i-th optimization variable, $c_i \in \mathbb{R}^{n_i}$ is the i-th cost vector, $X_i \subset \mathbb{R}^{n_i}$ is the i-th polyhedral constraint set and $A_i \in \mathbb{R}^{S \times n_i}$ is a matrix for the i-th contribution to the *coupling constraint* $\sum_{i=1}^{N} A_i x_i \leq b \in \mathbb{R}^S$. Problem (14) enjoys the constraint-coupled structure [38] and can be recast into a master-subproblem architecture by using the so-called *primal decomposition* technique [39]. The right-hand side vector b of the coupling constraint is interpreted as a given (limited) resource to be shared among the network agents. Thus, local *allocation vectors* $y_i \in \mathbb{R}^S$ for all i are introduced such that $\sum_{i=1}^{N} y_i = b$. To determine the allocations, a *master problem* is introduced

$$\begin{aligned} \min_{y_1,\ldots,y_N} \quad & \sum_{i=1}^{N} p_i(y_i) \\ \text{subj. to} \quad & \sum_{i=1}^{N} y_i = b \\ & y_i \in Y_i, \qquad \forall i \in \mathbb{I}, \end{aligned} \tag{15}$$

where, for each $i \in \mathbb{I}$, the function $p_i : \mathbb{R}^S \to \mathbb{R}$ is defined as the optimal cost of the i-th (linear programming) *subproblem*

$$\begin{aligned} p_i(y_i) = \min_{x_i} \quad & c_i^\top x_i \\ \text{subj. to} \quad & A_i x_i \leq y_i \\ & x_i \in X_i. \end{aligned} \tag{16}$$

In problem (15), the new constraint set $Y_i \subseteq \mathbb{R}^S$ is the set of y_i for which problem (16) is feasible, i.e., such that there exists $x_i \in X_i$ satisfying the local *allocation constraint* $A_i x_i \leq y_i$. Assuming problem (14) is feasible and X_i are compact sets, if $(y_1^\star, \ldots, y_N^\star)$ is an optimal solution of (15) and, for all i, $x_i^\star$ is optimal for (16) (with $y_i = y_i^\star$), then $(x_1^\star, \ldots, x_N^\star)$ is an optimal solution of the original problem (14) (see, e.g., [39, Lemma 1]).

APPENDIX C
PROOFS

A. Proof of Lemma IV.1

Note that problem (4) is the epigraph form of

$$\min_{x_i,B_i,Q_i} \sum_{(j,k)\in\mathcal{E}_A} c_i^{jk} x_i^{jk} + M \max\left\{0, \max_{j\in R}\left([y_i^t]_j - \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk}\right)\right\}$$

subj. to $(x_i, B_i, Q_i) \in \text{conv}(Z_i)$

Moreover, it holds $\text{conv}(Z_i) \supset Z_i$. Therefore, the proof follows since Z_i is not empty (by assumption). □

B. Proof of Lemma IV.2

Fix a robot i. Because of the thresholding operation (6), it holds $[y_i^{\text{END}}]_j \leq 1$ for all $j \in R$. We now show that the feasible set of problem (7) is not empty. Since problem (1) is assumed to be feasible, we know that there exists $z_i = (x_i, B_i, Q_i) \in Z_i$. Thus, we only have to show that the constraint $\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq [y_i^{\text{END}}]_j$ for all $j \in R$ can be satisfied by at least one vector $z_i \in Z_i$.

Due to the flow constraints (1e), the subtour elimination constraints (1h) and the integer constraints (1g), for all $j \in R$ the quantity $\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk}$ is either equal to 0 or equal to 1, since there can be at most one index k satisfying $(j,k) \in \mathcal{E}_A$ and such that $x_i^{jk} = 1$. Consider the constraint $\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq [y_i^{\text{END}}]_j$ and fix a component $j \in R$. Note that this constraint essentially imposes whether or not robot i must pass through location j. Indeed, on the one hand, if $[y_i^{\text{END}}]_j \leq 0$, the vector z_i can be chosen such that the left-hand side is either equal to 0 (vehicle i does not pass through location j) or equal to 1 (vehicle i passes through location j). In either case, it holds $\sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq 0 \geq [y_i^{\text{END}}]_j$ so that the constraint is satisfied. On the other hand, if $0 < [y_i^{\text{END}}]_j \leq 1$ (recall that $[y_i^{\text{END}}] \leq 1$ for all $j \in R$ and thus there are no other possibilities), then the only way to satisfy the constraint is to have $x_i^{jk} = 1$ for some index k with $(j,k) \in \mathcal{E}_A$, in which this case we would obtain $1 = \sum_{k:(j,k)\in\mathcal{E}_A} x_i^{jk} \geq [y_i^{\text{END}}]_j > 0$. As a consequence, problem (7) admits as feasible solution any vector $(x_i, B_i, Q_i) \in Z_i$ representing a path passing through all the locations $j \in R$ and satisfying $[y_i^{\text{END}}]_j > 0$. □

REFERENCES

[1] N. Kamra, T. S. Kumar, and N. Ayanian, "Combinatorial problems in multirobot battery exchange systems," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 852–862, 2017.

[2] A. Ham, "Drone-based material transfer system in a robotic mobile fulfillment center," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 2, pp. 957–965, 2019.

[3] 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.

[4] X. Bai, M. Cao, W. Yan, and S. S. Ge, "Efficient routing for precedence-constrained package delivery for heterogeneous vehicles," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 1, pp. 248–260, 2019.

[5] P. Toth and D. Vigo, *The vehicle routing problem*. SIAM, 2002.

[6] S. N. Parragh, K. F. Doerner, and R. F. Hartl, "A survey on pickup and delivery models part II: Transportation between pickup and delivery locations," *Journal für Betriebswirtschaft*, vol. 58, no. 2, pp. 81–117, 2008.

[7] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, no. 1, pp. 1–11, 2013.

[8] U. Ritzinger, J. Puchinger, and R. F. Hartl, "A survey on dynamic and stochastic vehicle routing problems," *International Journal of Production Research*, vol. 54, no. 1, pp. 215–231, 2016.

[9] B. Coltin and M. Veloso, "Online pickup and delivery planning with transfers for mobile robots," in *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[10] M. Liu, H. Ma, J. Li, and S. Koenig, "Task and path planning for multi-agent pickup and delivery." in *AAMAS*, 2019, pp. 1152–1160.

[11] M. H. F. b. M. Fauadi, S. H. Yahaya, and T. Murata, "Intelligent combinatorial auctions of decentralized task assignment for agv with multiple loading capacity," *IEEJ Transactions on electrical and electronic Engineering*, vol. 8, no. 4, pp. 371–379, 2013.

[12] B. Heap and M. Pagnucco, "Repeated sequential single-cluster auctions with dynamic tasks for multi-robot task allocation with pickup and delivery," in *German Conference on Multiagent System Technologies*. Springer, 2013, pp. 87–100.

[13] A. Arsie, K. Savla, and E. Frazzoli, "Efficient routing algorithms for multiple vehicles with no explicit communications," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2302–2317, 2009.

[14] A. Soeanu, S. Ray, M. Debbabi, J. Berger, A. Boukhtouta, and A. Ghanmi, "A decentralized heuristic for multi-depot split-delivery vehicle routing problem," in *2011 IEEE International Conference on Automation and Logistics (ICAL)*. IEEE, 2011, pp. 70–75.

[15] 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.

[16] 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.

[17] 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.

[18] A. Testa and G. Notarstefano, "Generalized assignment for multi-robot systems via distributed branch-and-price," *IEEE Transactions on Robotics*, 2021.

[19] L. Luo, N. Chakraborty, and K. Sycara, "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.

[20] A. Testa, A. Rucco, and G. Notarstefano, "Distributed mixed-integer linear programming via cut generation and constraint exchange," *IEEE Transactions on Automatic Control*, vol. 65, no. 4, pp. 1456–1467, 2019.

[21] 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.

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

[23] M. Pavone, N. Bisnik, E. Frazzoli, and V. Isler, "A stochastic and dynamic vehicle routing problem with time windows and customer impatience," *Mobile Networks and Applications*, vol. 14, no. 3, pp. 350–364, 2009.

[24] M. Pavone, E. Frazzoli, and F. Bullo, "Adaptive and distributed algorithms for vehicle routing in a stochastic and dynamic environment," *IEEE Transactions on automatic control*, vol. 56, no. 6, pp. 1259–1274, 2010.

[25] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith, "Dynamic vehicle routing for robotic systems," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1482–1504, 2011.

[26] A. Farinelli, A. Contini, and D. Zorzi, "Decentralized task assignment for multi-item pickup and delivery in logistic scenarios," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 1843–1845.

[27] 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.

[28] M. Saleh, A. Soeanu, S. Ray, M. Debbabi, J. Berger, and A. Boukhtouta, "Mechanism design for decentralized vehicle routing problem," in *Pro-*

ceedings of the 27th Annual ACM Symposium on Applied Computing, 2012, pp. 749–754.

[29] A. Falsone, K. Margellos, and M. Prandini, "A distributed iterative algorithm for multi-agent milps: finite-time feasibility and performance characterization," *IEEE control systems letters*, vol. 2, no. 4, pp. 563–568, 2018.

[30] A. Camisa, I. Notarnicola, and G. Notarstefano, "Distributed primal decomposition for large-scale MILPs," *IEEE Transactions on Automatic Control*, pp. 1–1, 2021.

[31] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*. Academic press, 1982.

[32] A. Testa, A. Camisa, and G. Notarstefano, "ChoiRbot: A ROS 2 toolbox for cooperative robotics," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2714–2720, 2021.

[33] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[34] 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.

[35] S. Wilson, P. Glotfelter, L. Wang, S. Mayya, G. Notomista, M. Mote, and M. Egerstedt, "The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems," *IEEE Control Systems Magazine*, vol. 40, no. 1, pp. 26–44, 2020.

[36] J.-F. Cordeau, "A branch-and-cut algorithm for the dial-a-ride problem," *Operations Research*, vol. 54, no. 3, pp. 573–586, 2006.

[37] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[38] G. Notarstefano, I. Notarnicola, and A. Camisa, "Distributed optimization for smart cyber-physical networks," *Foundations and Trends® in Systems and Control*, vol. 7, no. 3, pp. 253–383, 2019.

[39] G. J. Silverman, "Primal decomposition of mathematical programs by resource allocation: I–basic theory and a direction-finding procedure," *Operations Research*, vol. 20, no. 1, pp. 58–74, 1972.

Andrea Camisa received the Laurea degree summa cum laude in Computer Engineering from the University of Salento, Italy in 2017 and the Licenza degree from the excellence school ISUFI, Italy in 2018. He is a Ph.D student within the Ph.D Programme "Biomedical, Electrical, and Systems Engineering" at the Department of Electrical, Electronic and Information Engineering, University of Bologna, Italy. He was a visiting student at the University of Stuttgart in 2017 and 2018. His research interests include convex, distributed and mixed-integer optimization.

Andrea Testa received the Laurea degree "summa cum laude" in Computer Engineering rom the Università del Salento, Lecce, Italy in 2016 and the Ph.D degree in Engineering of Complex Systems from the same university in 2020.

He is a Research Fellow at Alma Mater Studiorum Università di Bologna, Bologna, Italy. 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.