Auction-Based Task Allocation and Motion Planning for Multi-Robot Systems with Human Supervision

Journal of Intelligent & Robotic Systems (2023) 109:24
https://doi.org/10.1007/s10846-023-01935-x

REGULAR PAPER

Auction-Based Task Allocation and Motion Planning for Multi-Robot Systems with Human Supervision

Giada Galati[1] · Stefano Primatesta[2] · Alessandro Rizzo[1]

Received: 13 February 2023 / Accepted: 18 July 2023

Abstract
This paper presents a task allocation strategy for a multi-robot system with a human supervisor. The multi-robot system consists of a team of heterogeneous robots with different capabilities that operate in a dynamic scenario that can change in the robots' capabilities or in the operational requirements. The human supervisor can intervene in the operation scenario by approving the final plan before its execution or forcing a robot to execute a specific task. The proposed task allocation strategy leverages an auction-based method in combination with a sampling-based multi-goal motion planning. The latter is used to evaluate the costs of execution of tasks based on realistic features of paths. The proposed architecture enables the allocation of tasks accounting for priorities and precedence constraints, as well as the quick re-allocation of tasks after a dynamic perturbation occurs –a crucial feature when the human supervisor preempts the outcome of the algorithm and makes manual adjustments. An extensive simulation campaign in a rescue scenario validates our approach in dynamic scenarios comprising a sensor failure of a robot, a total failure of a robot, and a human-driven re-allocation. We highlight the benefits of the proposed multi-goal strategy by comparing it with single-goal motion planning strategies at the state of the art. Finally, we provide evidence for the system efficiency by demonstrating the powerful synergistic combination of the auction-based allocation and the multi-goal motion planning approach.

Keywords Auction · Task allocation · Motion planning · Path planning · Multi-robot system · Multi-robot task allocation

1 Introduction

In the near future, multi-robot systems (MRSs) are envisaged to significantly impact different social fields [1]; from surveillance missions [2] to industrial applications [3], from rescue operations [4] to agriculture [5].

MRSs exhibit significant advantages over Single-Robot Systems (SRSs), due to their redundancy, flexibility, efficiency, and the absence of a single point of failure [6, 7]. However, communication, coordination, and control overhead are required in order to orchestrate the action of the team as a whole.

Complex and life-critical tasks, such as rescue operations, often involve the adoption of MRSs consisting of unmanned vehicles and human operators, where humans are in charge of important decisions and of some aspects of the coordination of the operations, especially those related to the evaluation of the overall success of the work plan, the safety of human lives, and the management of unforeseen situations. Rescue operations, in particular, involve a large quota of human operators within the team, which may attain up to two humans for each robot [8].

Reducing the number of human operators in such teams is desirable to enhance safety, avoid confounding factors emerging from the adoption of contrasting strategies by different operators in the team, and reduce the odds of human mistakes [9]. On the other hand, operations that involve ethical challenges related to *decision-making* [10] and *responsibility* on decision [11] will continue to require human intervention or supervision in the foreseeable future [12].

For example, choices about allocating resources in emergency situations, including where to concentrate rescue efforts, assessing risks, determining the order of people to

✉ Alessandro Rizzo
alessandro.rizzo@polito.it

[1] Department of Electronics and Telecommunications, Politecnico Di Torino, Corso Duca Degli Abruzzi 24, 10129 Torino, Italy

[2] Department of Mechanical and Aerospace Engineering, Politecnico Di Torino, Corso Duca Degli Abruzzi 24, 10129 Torino, Italy

Published online: 16 September 2023

be rescued, prioritizing medical treatment, managing who must be left to wait, and optimizing the utilization of scarce resources are unplausible to be made by teams constituted by robots only [10]. In this context, Harbers et al. [11] raise the issue related to moral and legal responsibility, where the former concerns blame and the latter concerns accountability. These issues, according to the authors, occur when robots are not supervised by a human. If a robot undergoes a malfunctioning, behaves inappropriately, makes an error, or causes harm, it can be difficult to determine who is responsible for the resulting damage. This issue becomes even more complex when the robot has some level of autonomy, self-learning abilities, or is capable of making decisions that were not explicitly programmed.

This calls for the design of robust and efficient design of coordination algorithms for MRSs with human supervisors, referred as to HMRS in the following.

In this work, we propose a task allocator strategy for a HMRS with heterogeneous capabilities, where the human supervisor can be either a pilot of one of the robots or an external coordinator. The proposed task allocation can manage a dynamic environment, involving both changes in the operation requirements or in the robots' capabilities. Also, a human supervisor can intervene in the planning process by: *(i)* approving or canceling a proposed plan; or *(ii)* introducing new constraints to a proposed plan; for example, by assigning a specific task to a given robot, along with a given execution time set for safety reasons or due to a change in the capabilities required to execute a given task. We advance the state of the art along two main directions. First, our task allocation combines an auction-based strategy [13] with a motion planner [14] enhanced with a multi-goal approach, to take full advantage of the features of the sequential single-item auction and leverage real and measurable features of the path to be accomplished, rather than its mere description. Second, flexibility in operations is attained through dynamic re-allocation, which can be triggered at any time, either by changes in the operational conditions or by the human supervisor.

1.1 Previous Works

Our contribution falls in the broad category of multi-robot task allocation problems (MRTA) [15] –a variant of the multiple Traveling Salesman Problem (mTSP) [15], which is notoriously NP-hard.

Main approaches to the solution of task allocation problems are Mixed-Integer Linear Programming (MILP) [16, 17], and auction-based techniques [2, 13, 18–22].

The former may lead toward the optimal solution at the cost of an often unaffordable computational complexity, which calls for the combined usage of heuristics and the consequent attainment of suboptimal solutions. In the context of HMRS, the support of a human supervisor was included in [17] to evaluate the intermediate solutions of the MILP based on objective or subjective quality criteria and personal expertise. In this way, also sub-optimal solutions may be adopted, and the solver can be conducted to an early termination. However, in this case, the human supervisor is continuously required to evaluate operational scenarios, practically providing heuristic criteria to reduce the computational load of the solver, at the expense of their own cognitive load, entailing an increase in the level of stress and subtracting precious intellectual resources to the execution of complex tasks.

The latter, on the other hand, consists of an iterative strategy based on the optimization of the interest of selfish agents, typically leading to sub-optimal solutions with a reasonable computational complexity. Auction algorithms have grown in popularity within the robotics community [13] to handle task allocation problems [2, 23] efficiently and robustly [18]. In an auction, each robot (the bidder) places a bid to commit to the execution of each task (item) based on a given cost function. Then, a coordinator (the auctioneer) assigns (sells) the items to the highest or the lowest bidder, depending on whether the considered cost function should be maximized or minimized [13]. Auctions are particularly suitable for dynamically-changing environments and can be deployed in centralized, decentralized, and distributed architectures [24]. Specifically, the calculations of the auctioneer and the bidders can be done on a single system (centralized), multiple systems (decentralized), or without a unique and centralized auctioneer (distributed).

In the literature, auction-based methodologies have been used in different applications with MRS, such as exploration and destruction, patrolling, and surveillance mission, [2, 18, 25]. Notably, [2] adopts an auction-based methodology to solve a task allocation problem of a HMRS in a dynamic scenario with priority constraints between tasks. Differently from our approach, however, the human-controlled vehicle has neither supervisory features nor specific privileges. This makes the solution of the problem equivalent to that computed for a fully automated team.

Some works in the literature aimed at combining a task allocation strategy with a motion planning. In [26] a MILP is combined with an RRT*-based algorithm, while in [27] an integer programming model integrates a motion planner based on a genetic algorithm. Instead, authors in [21, 28] integrate an auction-based task allocator with the A* algorithm. Specifically, in [21], the authors apply the auction in a dynamic environment for UAVs where the mission is continuously allocated and executed autonomously.

Auction-based task allocation is also combined with RRT-based algorithms [29, 30]. RRTs are suitable for supporting task allocation because they are able to rapidly compute a path in the search space by constructing an incremental

exploration tree [31]. However, studies in [29, 30] use the standard RRT algorithm, which has the drawback of computing non-optimal solutions. The optimality of the motion planning is an essential feature for the quality of the task allocation because the computed paths are evaluated to assign the task to the robot that offers the best solution. For this reason, differently from [29, 30], our motion planner guarantees an optimal path thanks to the RRT$^{\#}$ algorithm [14].

1.2 Our Contributions

The HMRS aims to handle a complex operation happening in a dynamic environment. We assume that such an operation may be decomposed according to a hierarchical structure, illustrated in Fig. 1.

Such a complex operation consists of some independent sub-operations that must be executed by a robot with appropriate capabilities. Each sub-operation may have a priority. Each sub-operation is in turn composed of several tasks, subject to precedence constraints.

The operation structure mentioned above is relevant to many complex operation scenarios, such as people rescuing. In this context, the operation consists of some sub-operations equal to the number of people to be rescued. In particular, each sub-operation consists of all the actions (tasks) necessary to save one person (target). Each target has a priority that is related to the urgency of the rescue. Each task coincides with the visit of a location in the operational scenario.

The HMRS consists of a team of heterogeneous robots, i.e. each robot has a set of capabilities, which allow it to execute certain tasks. The dynamic nature of the environment where the HMRS operates may elicit re-allocation upon changes in the operational conditions, also called *perturbations*. For instance, a robot or one of its capabilities may become unavailable due to a collision, a system failure, the exhaustion of its battery; or the human supervisor demands a re-allocation, due to safety reasons or other technical considerations not intelligible by machines. More specifically, the human supervisor may trigger a re-allocation of the HMRS through one of the following actions: *(i)* rejecting the computed plan before its execution; or, *(ii)* forcing a particular robot to execute a task. Regarding the rescue operation, the former is typically related to an overall approval of the plan, in light of ethical or safety implications, while the latter may relate to on-the-go decisions that may increase the chances of safety in light of the actual operational conditions. Such a dynamic scenario is summarized in the flow chart of Fig. 2.

Our effort presents several novelties and improvements compared to the state of the art. First, we propose an auction-based method for a heterogeneous team operating in a dynamic scenario with human supervision, which supports precedence constraints between tasks, priority between sub-operations, and on-the-go re-allocation due to perturbations, coming either from the environment or from the human supervisor –a setting that was not entirely contemplated in the past [2, 17, 18, 20, 21, 25]. Along the lines of [17, 32], we design a system able to simulate the intervention of the human supervisor by dynamically adding constraints to the auction-based task allocation, such as forcing a robot to execute a specific task within a given completion time or changing the capability required for a given task. This scenario may arise for safety reasons or, for example, to adjust the allocation problem when a malfunction occurs, thus enabling the operation to be completed. However, the MILP approach used in [17] hampers its concrete applicability to complex and dynamically changing scenarios, due to its inherent computational burden. Conversely, Hussaini et al in [32] describe a scenario where the multi-robot system is supervised by a human operator who can actively address corrective actions in the assignment plan based on the estimated or the notified contingencies. However, their re-allocation process is handled by using a heuristic-based task allocation, which may face scalability issues, making it inefficient or impractical to find a suitable allocation within a reasonable time.

Second, along the lines of [33], we use a multi-goal motion planner in combination with the auction-based allocation, achieving an overall method that is fast, effective, and reactive to perturbations. However, in [33], the authors are focused more on the optimality of the planning rather than on the responsiveness of the whole system. Also, the efficiency of the strategy claimed in [33] is hampered by the assignment

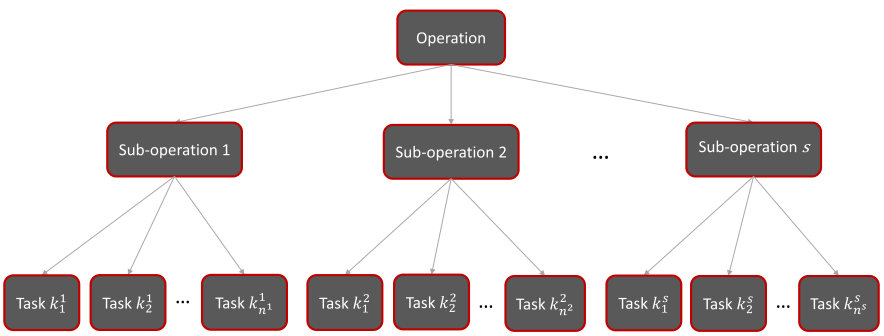

Fig. 1 Hierarchical structure of operation, sub-operations, and tasks

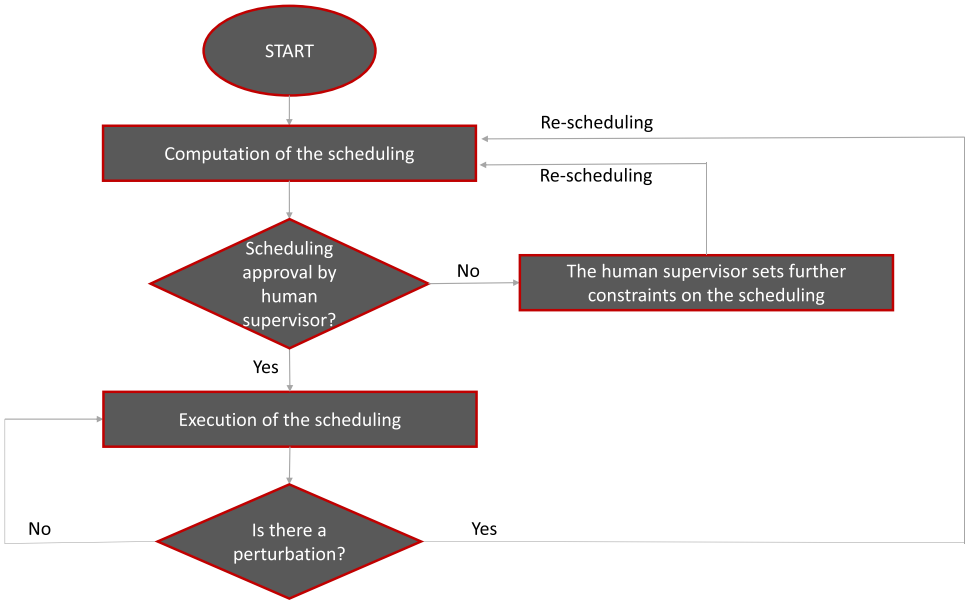

Fig. 2 Dynamic scenario of an operation and re-allocation mechanisms

of capacity constraints to each robot. Notably, with more than six scheduled tasks for each robot, the computational burden already tends to become unmanageable. Here, such constraints are not posed and efficiency is privileged, if necessary, through a trade-off between computational burden and pursuit of optimality. Moreover, in [33] the authors use the general logic of the auction but the motion planner does not leverage any particular feature of the auction to work in synergy with it. In fact, the motion planner creates a general graph that is used to evaluate the effect of the candidate task on the entire robot schedule. In particular, the graph is useful to evaluate different links (robot-task and task-task) and finally to compute the best solution for the TSP problem. Here, on the other hand, the multi-goal motion planner is used to fully leverage the features of the sequential single-item auction by simultaneously computing the cost for each robot to accomplish a given task.

To the best of our knowledge, there are no works that describe a system incorporating a human supervisor with the aforementioned functions, imposing constraints on an auction-based task allocator with the described features, specifically designed to address ethical challenges in demanding environments.

Moreover, we remark that, although the strategy proposed in this paper is tailored to a rescue scenario for illustrative purposes, the application field is extensive and may embrace robots of heterogeneous nature, such as ground, aerial, or underwater.

The rest of the paper is structured as follows. Section 2 explains the problem statement and states the assumptions of our approach. Section 3 presents our methodology, based on an auction-based task allocator and a multi-goal RRT$^{\#}$ algorithm. The effectiveness and robustness of our methodology are demonstrated by simulations in Section 4. Finally, in Section 5, we draw our conclusions and offer a discussion toward further developments.

2 Problem Statement

In the following, we use roman font to denote scalar quantities ($x \in \mathbb{R}$), low bold font to denote vectors ($\boldsymbol{x} \in \mathbb{R}^2$), and upper bold font to denote sets ($\boldsymbol{X} \in \mathbb{R}^N$) and matrices ($\boldsymbol{X} \in \mathbb{R}^{N \times M}$).

We assume a two-dimensional operational space $\boldsymbol{X} \in \mathbb{R}^2$ defined as a Euclidean state space in which each element $\boldsymbol{x} \in \boldsymbol{X}$ represents a possible location for a robot. The subset $\boldsymbol{X}_{\mathbf{obs}} \subseteq \boldsymbol{X}$ contains locations where a robot cannot be located, e.g. those occupied by obstacles. We assume that the positions of the obstacles are known a-priori to the task allocator and the motion planner. The set $\boldsymbol{X}_{\mathbf{free}} = \boldsymbol{X} \setminus \boldsymbol{X}_{\mathbf{obs}}$ includes the remaining positions where a robot can be located, also called the *valid* locations.

The HMRS comprises m robots and is identified by the set $\boldsymbol{R} = \{r_1, r_2, \ldots, r_m\}$. The set $\boldsymbol{X}_{\mathbf{r}} = \{\boldsymbol{x}(r_1), \boldsymbol{x}(r_2), \ldots, \boldsymbol{x}(r_m)\}$ indicates the position of each robot, with $\boldsymbol{x}(r_i) \in \boldsymbol{X}_{\mathbf{free}}$, $i = 1, \ldots, m$.

The set $\boldsymbol{Cap} = \{p_1, p_2, \ldots, p_l\}$ indicates the l available capabilities used to execute all the tasks by the multi-robot system. A capability is a particular feature that empowers a robot to accomplish a particular operation; for example, the capability of moving hazardous materials or illuminating the scene at night.

Each robot has different capabilities that may change in time. They are summarized in a boolean time-varying matrix $\boldsymbol{RC}(\boldsymbol{t})$ of dimensions $m \times l$. The element $RC(t)_{i,j}$ is set to

one if the robot i, with $i = 1, \ldots, m$, is equipped with the capability j, with $j = 1, \ldots, l$, and to zero otherwise.

The operation to be allocated aims to manage s targets defined by the set $\boldsymbol{G} = \{g_1, g_2, ..., g_s\}$. The set $\boldsymbol{X_g} = \{\boldsymbol{x}(g_1), \boldsymbol{x}(g_2), \ldots, \boldsymbol{x}(g_s)\}$ indicates the position of each target $g \in \boldsymbol{X_{free}}$. Each target has a priority defined by the set $\boldsymbol{GP} = \{gp_1, gp_2, ..., gp_s\}$ that defines which target has to be managed first, with $gp_i \in \mathbb{N}, i = 1, \ldots, s$.

In particular, the higher the priority, the more urgent the target to manage. Nevertheless, it might also happen that two or more sub-operations have the same priority, then the auction will try to handle them in parallel, when possible.

Hence, the operation consists of s sub-operations because each sub-operation is responsible for managing only one target while respecting its priorities.

Each sub-operation consists of several tasks. The set $\boldsymbol{K^i} = \left\{k_1^i, k_2^i, \ldots, k_{n^i}^i\right\}$ denotes the list of n^i tasks that form the sub-operation i, with $i = 1, 2, ..., s$, in which the subscript represents the sequencing of the tasks. Tasks must be performed sequentially. For instance, task k_2^i has to be performed after the task k_1^i. $\boldsymbol{K^{tot}} = \boldsymbol{K^1} \cup \boldsymbol{K^2} \cup \ldots \cup \boldsymbol{K^s}$ represents the set of all tasks, with cardinality $n^{\text{tot}} = n^1 + n^2 + ... + n^s$.

Each task has to be performed in a specific location. The set $\boldsymbol{X_k} \subseteq \boldsymbol{X_{free}}$ includes the positions of the free space, where all tasks must be executed. The notation $\boldsymbol{x}(k_j^i) \in \boldsymbol{X_k} \subseteq \boldsymbol{X_{free}}$ indicates the position of a task k_j of the sub-operation $i = 1, ..., s$. We assume that the task allocator and the centralized motion planner know the positions of every robot $\boldsymbol{x}(r_i) \in \boldsymbol{X_r}$ and every task $\boldsymbol{x}(k_j^i) \in \boldsymbol{X_k}$.

The subdivision of the operation in sub-operations and, subsequently, in tasks is shown in Fig. 1. The decomposition of the complex operation in its tasks is out of the scope of this paper; hence, we assume that the sets of sub-operations and tasks are made available to the task allocator by an external mechanism.

Each task requires some capabilities to be performed. The combination of tasks and capabilities is summarized in a boolean matrix $\boldsymbol{TC}$ of dimensions $n^{\text{tot}} \times l$. Element $TC_{i,j}$ is set to one if the task i, with $i = 1, 2, \ldots, n^{\text{tot}}$ requires the capability j, with $j = 1, 2, \ldots, l$; it is set to zero otherwise.

The role of the task allocation is to handle the n^{tot} tasks to the HMRS composed by m robots equipped with different capabilities l. The computed plan is designed to optimize the total time of the operation, guaranteeing that tasks are executed by the robots that possess proper capabilities, respecting the prioritization between sub-operations and precedence constraints between tasks. Re-allocation can be triggered by *perturbations,* which can be *external* or *internal*.

An external perturbation is caused by an external and unexpected event, such as a system or sensor failure which can cause the loss of a robot or the loss of its capabilities.

An internal perturbation occurs when it is caused by an internal event, e.g. a change of strategy forced by the human supervisor, such as changing capability for a given task or assigning a task to a particular robot.

In the following, the term t_{new} defines the time instant when a perturbation occurs considering a continuous time.

The intervention of the human supervisor is defined by the boolean matrix $\boldsymbol{TC^H}$ of dimension $n^{\text{tot}} \times l$, in which each element $TC_{i,j}^{\text{H}}$ defines if the human supervisor forces the capability j, with $j = 1, 2, \ldots, l$ to perform the task i, with $i = 1, 2, \ldots, n^{\text{tot}}$. Instead, the matrix $\boldsymbol{C_p}$ of dimension $n^{\text{tot}} \times m$ includes the completion time forced by the human supervisor. Each element $C_{\text{p}}(i, j)$ defines to whom the task i, with $i = 1, 2, \ldots, n^{\text{tot}}$ is assigned to the robot j, with $j = 1, 2, \ldots, l$ and when the task i must be completed.

3 Methodology

In this paper, a centralized approach is adopted since the human supervisor must have the possibility to approve the final plan and to take action (e.g. change capabilities for a given task or assign a task to a particular robot) about the plan in two different situations: when the plan is in execution; and when the human supervisor does not approve the plan.

Once the plan is approved, assigned tasks are executed in a completely autonomous fashion. That is, each robot is able to move toward the assigned position and autonomously execute its task, counting only on its capabilities. We also assume that, once scheduled, each robot is able to perform the planned tasks successfully.

The centralized system is composed by a task allocator based on a sequential single-item auction, and a centralized motion planner based on RRT$^{\#}$ with a multi-goal approach. These blocks continuously interact to compute all the paths connecting robots and tasks and estimate their costs in order to compute the plan which will be checked by the human supervisor, as shown in Fig. 3.

The communication between the two blocks is assumed as *ideal* –without delays and losses of information.

In the following section, each algorithm is detailed.

3.1 Algorithms

3.1.1 Auction-Based Task Allocator

A traditional auction is composed of two steps: the bidding step and the winner determination step. In the *bidding step*, the auctioneer informs the robots about the tasks for sale. Then, each robot evaluates the tasks, calculates the bid, and returns the bid to the auctioneer. Then, during the *winner*

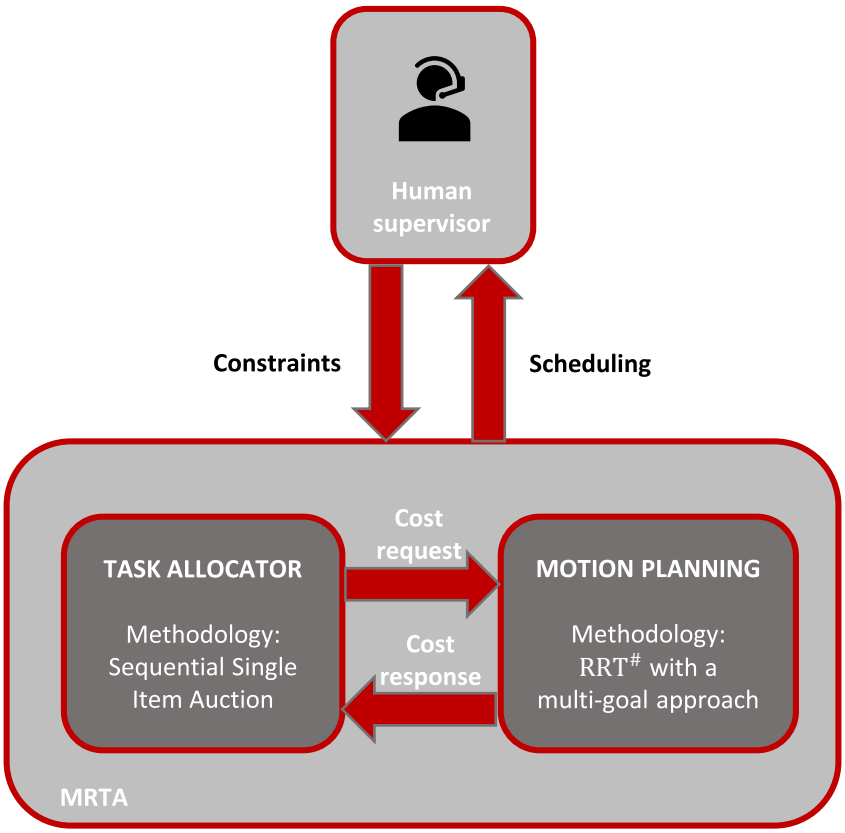

Fig. 3 Overview of the methodology for each block of the HMRTA

determination step, the auctioneer determines the winner for each task and informs the winning robots. These two steps compose the so-called *round* of the auction.

In our problem, we choose a sequential single item (SSI) auction where the auctioneer (the centralized task allocator) sells one item (task) for each round in an order selected respecting the priority of targets s.

In the proposed strategy, a centralized motion planner is called at each round with the aim of computing the bids of all robots to perform a task. In fact, the bid returned by a robot includes the cost of moving toward the task's position.

During the winner determination step, the auctioneer assigns the task to the robot r_{best} with the right capabilities and with the lowest bidder.

In particular, our algorithm based on a sequential single-item auction with the decision of the human supervisor is summarized in Algorithm 1.

The **inputs** of the task allocator are: the set of all the tasks $\boldsymbol{K}^{\mathbf{tot}}$, the set of robots $\boldsymbol{R}$, the possible instant of perturbation t_{new}, the set of priority for each sub-operation $\boldsymbol{GP}$, the matrix with the combination of robots and capabilities $\boldsymbol{RC}(t)$, the matrix with the combination of tasks and capabilities $\boldsymbol{TC}$, the operational space $\boldsymbol{X}$ including obstacles and free space, the vector of robots' positions $\boldsymbol{X}_{\mathbf{r}}$, the vector of tasks' positions $\boldsymbol{X}_{\mathbf{k}}$, the matrix with the assignment of tasks to robots made by the human supervisor $\boldsymbol{C}_{\mathbf{p}}$, and the matrix $\boldsymbol{TC}^{\mathbf{H}}$ with the assignment of sensors to tasks made by the human supervisor.

The Algorithm 1 is split into two macro steps: **Initialization** and **Auction**.

The **Initialization** is fundamental in order to create and initialize variables essential for the auction.

$\boldsymbol{C}$ is the matrix of the completion time for all the tasks, where the element $C(k, r)$ denotes the completion time of the task $k \in \boldsymbol{K}^{\mathbf{tot}}$ performed by the robot $r \in \boldsymbol{R}$ (line 3). $\boldsymbol{T_0}$ is the vector of the starting times of all the tasks n^{tot}, where $t_0(k)$ is the starting time of task $k \in \boldsymbol{K}^{\mathbf{tot}}$ (line 4).

$\boldsymbol{TC}^{\mathbf{H}}$ is the matrix with the capabilities assigned to the tasks by the human supervisor. Each element $TC^{\text{H}}_{i,j}$ defines if the task i, with $i = 1, 2, ..., n^{\text{tot}}$ requires the capability j, with $j = 1, 2, ..., l$. If the $\boldsymbol{TC}^{\mathbf{H}}$ matrix is empty, the human supervisor has not added any constraint on the capabilities for the tasks. Otherwise, the function **HumanChoiceCapabilities** updates the $\boldsymbol{TC}$ matrix with the information of $\boldsymbol{TC}^{\mathbf{H}}$ (line 6).

The **Auction** represents the main task allocation algorithm. In this macro step, if at least a robot performing each task exists, the auction handles sequentially each task according to the list of prioritized tasks.

In the following, we describe each function of Algorithm 1:

Algorithm 1 Task allocation algorithm based on Auction

1 **Input**: $\boldsymbol{K}^{\mathbf{tot}}$, $\boldsymbol{R}$, t_{new}, $\boldsymbol{GP}$, $\boldsymbol{RC}(t)$, $\boldsymbol{TC}$, $\boldsymbol{X}$, $\boldsymbol{X}_{\mathbf{obs}}$, $\boldsymbol{X}_{\mathbf{r}}$, $\boldsymbol{X}_{\mathbf{k}}$, $\boldsymbol{C}_{\mathbf{p}}$, $\boldsymbol{TC}^{\mathbf{H}}$
2 **Initialization**:
3 $\boldsymbol{C} \leftarrow zeros(n^{\text{tot}}, m)$
4 $\boldsymbol{T_0} \leftarrow zeros(n^{\text{tot}})$
5 **if** $\boldsymbol{TC}^{\mathbf{H}}$ *is not empty* **then**
6 $\quad \boldsymbol{TC} \leftarrow$ HumanChoiceCapabilities($\boldsymbol{TC}^{\mathbf{H}}$)
7 **Auction**:
8 $\boldsymbol{M}_{\mathbf{st}} \leftarrow$ CreationStaticMask($\boldsymbol{TC}$, $\boldsymbol{RC}$)
9 **if** ControllingFeasibleOperation($\boldsymbol{M}_{\mathbf{st}}$) = True **then**
10 $\quad \boldsymbol{L}_{\mathbf{pr}} \leftarrow$ CreationListPrioritizedTask($\boldsymbol{K}^{\mathbf{tot}}$, $\boldsymbol{GP}$)
11 $\quad (\boldsymbol{K}_{\mathbf{forced}}, \boldsymbol{C}) \leftarrow$ HumanChoiceTasksRobot($\boldsymbol{C}_{\mathbf{p}}$, $\boldsymbol{L}_{\mathbf{pr}}$)
12 $\quad$ **foreach** *task* $k_{\text{pr}} \in \boldsymbol{L}_{\mathbf{pr}}$ **do**
13 $\quad\quad$ **if** $k_{\text{pr}} \notin \boldsymbol{K}_{\mathbf{forced}}$ **then**
14 $\quad\quad\quad (\boldsymbol{M}_{\mathbf{dyn}}, t_0^{\text{exp}}(k_{\text{pr}})) \leftarrow$ CreationDynamicMask($\boldsymbol{M}_{\mathbf{st}}$, $\boldsymbol{C}$, $\boldsymbol{K}^{\mathbf{tot}}$, k_{pr}, t_{new})
15 $\quad\quad\quad$ (**Costs**, **Times**) $\leftarrow$ GetCosts($\boldsymbol{M}_{\mathbf{st}}$, $\boldsymbol{X}$, $\boldsymbol{X}_{\mathbf{r}}$, $\boldsymbol{x}(k_{\text{pr}})$)
16 $\quad\quad\quad$ **if** ControllingAvailabilityRobots(k_{pr}, $\boldsymbol{M}_{\mathbf{dyn}}$) = False **then**
17 $\quad\quad\quad\quad r_{\text{best}} \leftarrow$ SelectionBestRobot(**Costs**, $\boldsymbol{M}_{\mathbf{st}}$)
18 $\quad\quad\quad\quad t_0(k_{\text{pr}}) \leftarrow \max(\text{C(k, r}_{\text{best}})) \quad \forall k \in \boldsymbol{K}^{\mathbf{tot}}$
19 $\quad\quad\quad$ **else**
20 $\quad\quad\quad\quad r_{\text{best}} \leftarrow$ SelectionBestRobot(**Costs**, $\boldsymbol{M}_{\mathbf{dyn}}$)
21 $\quad\quad\quad\quad t_0(k_{\text{pr}}) \leftarrow t_0^{\text{exp}}(k_{\text{pr}})$
22 $\quad\quad\quad C(k_{\text{pr}}, r_{\text{best}}) \leftarrow t_0(k_{\text{pr}}) + Times(r_{\text{best}})$
23 $\quad\quad\quad \boldsymbol{x}(r_{\text{best}}) \leftarrow \boldsymbol{x}(k_{\text{pr}})$
24 **else**
25 $\quad$ **return** warning to supervisor

- CreationStaticMask: the main goal of this function is to create a static mask that defines which robot is able to do which task. In particular, the $\boldsymbol{M}_{\mathbf{st}}$ is a boolean matrix, where the element $M_{\text{st}}(k, r)$ denotes if the robot r is able to perform the task k;
- ControllingFeasibleOperation: given the static mask $\boldsymbol{M}_{\mathbf{st}}$, this function controls if at least a robot is able to perform each task. If not, the task allocation cannot solve the problem and the function returns a **False** state, warning the supervisor (line 25). Otherwise, the auction can be performed;
- CreationListPrioritizedTask: this function computes the list $\boldsymbol{L}_{\mathbf{pr}}$, in which each task is ordered sequentially starting with the one with the highest priority. If two tasks have the same priority, then the algorithm randomly chooses the task to be evaluated first. This situation could happen when there are sub-operations with the same priority;
- HumanChoiceTasksRobot: given the matrix with the assignment of tasks to robots $\boldsymbol{C}_{\mathbf{p}}$ forced by the human supervisor and the list of the prioritized tasks $\boldsymbol{L}_{\mathbf{pr}}$, this function updates the matrix of the completion time $\boldsymbol{C}$ and computes the vector of the tasks already assigned by the human supervisor $\boldsymbol{K}_{\mathbf{forced}}$;
- CreationDynamicMask: if the selected task k_{pr} is not located in $\boldsymbol{K}_{\mathbf{forced}}$ (line 13), the task k_{pr} has not already been allocated and, then, the auction tries to assign the task. Given the static mask $\boldsymbol{M}_{\mathbf{st}}$, the matrix of the completion time $\boldsymbol{C}$, the lists of sub-operations with the corresponding sequences between tasks $\boldsymbol{K}_{\mathbf{tot}}$, the task to be handled k_{pr}, and the eventual instant of perturbation t_{new} (if we are in the re-allocation phase), this function computes the time in which the task k_{pr} should start $t_0^{\text{exp}}(k_{\text{pr}}) \in \boldsymbol{T}_0^{\mathbf{exp}}$ and the dynamic mask $\boldsymbol{M}_{\mathbf{dyn}}$. $\boldsymbol{M}_{\mathbf{dyn}}$ is a boolean matrix that allows the algorithm to know which robot is busy when the algorithm is assigning the task k_{pr} ($t_0^{\text{exp}}(k_{\text{pr}})$) and does not have the capabilities to perform the task k_{pr}. For completeness, the dimensions of the dynamic mask $\boldsymbol{M}_{\mathbf{dyn}}$ are the same as the static mask $\boldsymbol{M}_{\mathbf{st}}$;
- GetCosts: this function provides the interaction with the motion planner implementing the *bidding* step of the auction. Given the static mask $\boldsymbol{M}_{\mathbf{st}}$, the operational space $\boldsymbol{X}$, the robots' positions $\boldsymbol{X}_{\mathbf{r}}$, the task position $\boldsymbol{x}(k_{\text{pr}})$, the motion planning computes the costs (**Costs**) and execution times (**Times**) to reach the task $\boldsymbol{x}(k_{\text{pr}})$ by each robot that has the proper capabilities. In our problem, we solely consider the time required to reach the position of a task, as we assume that the execution time of the task is typically negligible than the time to reach its position. More details about this function have been provided below with the description of Algorithm 2;
- ControllingAvailabilityRobots: this function controls if at least one robot available to perform the task k_{pr} at the

expected starting time $t_0^{\exp}(k_{\text{pr}})$ exists by checking the dynamic mask $\boldsymbol{M}_{\mathbf{dyn}}$. If it does not exist, the function returns a **False** value. This condition implies that at the instant of assignment ($t_0^{\exp}(k_{\text{pr}})$) there is no free robot because robots that would have the capabilities to perform the task k_{pr} are busy;

– SelectionBestRobot: this function provides the second step of the auction– *the winner determination step*. If no robot can perform the task k_{pr} at the expected starting time $t_0^{\exp}(k_{pr})$ (i.e. ControllingAvailabilityRobots() = False), the SelectionBestRobot function selects the robot with the minimum cost to perform the task k but considering the static mask $\boldsymbol{M}_{\mathbf{st}}$. This detail is important because in this case, the choice of the best robot (r_{best}) is made only in consideration of who has the capabilities to do it and thus not considering the availability at the expected starting time $t_0^{\exp}(k_{\text{pr}})$. For this reason, the real starting time $t_0(k_{\text{pr}})$ for the task k_{pr} is updated considering the maximum value between the completion time of the tasks already assigned to the winner robot (line 18). On the other hand, if at least a robot to perform the task k_{pr} exists (i.e. ControllingAvailabilityRobots() = True), the SelectionBestRobot() function selects the robot with the minimum cost to perform the task k_{pr} but, unlike the previous case, considering the dynamic mask $\boldsymbol{M}_{\mathbf{dyn}}$ since we want to allocate the task at $t_0^{\exp}(k_{\text{pr}})$. Then, in line 21 the actual starting time ($t_0(k_{\text{pr}})$) for the task k_{pr} is updated with the expected one ($t_0^{\exp}(k_{\text{pr}})$). Finally, the completion time for task k_{pr} is computed (line 22), and the position of the winner robot is updated with the position of task k_{pr} (line 23).

3.1.2 Motion Planner

As previously defined, the motion planner algorithm is called several times by the task allocation algorithm with the function GetCosts. The motion planner is implemented using the RRT$^{\#}$ algorithm extended with a multi-goal strategy. In fact, in this work, the well-known RRT$^{\#}$ is exploited to construct an asymptotically optimal graph exploring the entire map (i.e. the search space). The graph is rooted from the task position and is constructed by randomly sampling and connecting states of the search space as in [14]. Hence, we use the constructed graph to compute all the paths connecting the task position with the robot positions. In fact, as with all the RRT-based algorithms, only one branch of the graph exists connecting the origin of the graph (i.e. the task position) and any other state of the graph. This strategy is perfectly suited to the centralized task allocator because only one exploration graph is constructed to compute all the paths and their costs, instead of computing all the paths sequentially as commonly performed in the literature. An example of this strategy is shown in Fig. 4.

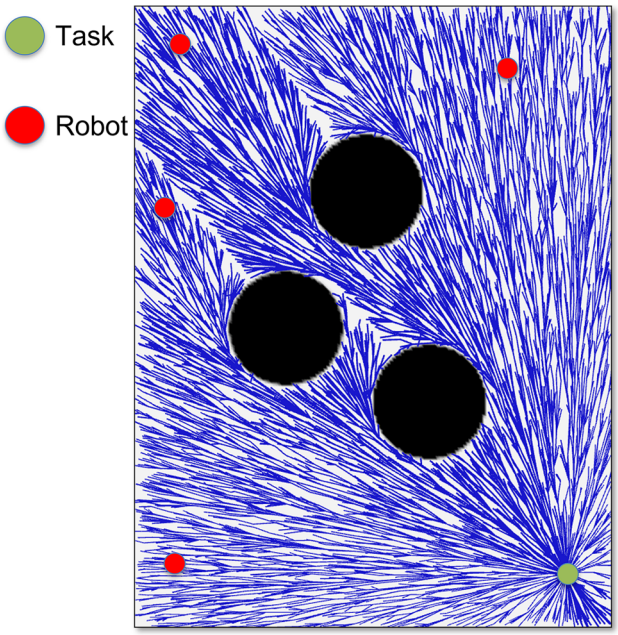

Fig. 4 Example of the exploration graph constructed by the RRT$^{\#}$ algorithm rooted from the task position. The graph (in blue) explores the map reaching all the robots (in red) avoiding the obstacles (in black). The computed path per each robot is the branch connecting task and robot positions

The pseudocode of the motion planner is described in Algorithm 2. The inputs of the function are: the set $\boldsymbol{X}_{\mathbf{r}}$ with the robot positions; the position $\boldsymbol{x}(k_{\text{pr}})$ of the task k_{pr}; the matrix $\boldsymbol{M}_{\mathbf{st}}$ that defines which robots have the capabilities to execute the task k_{pr}; and the operational space $\boldsymbol{X}$ that determines the search space of the motion planning problem including obstacles.

First, the task position is added to the graph $\mathcal{G}$ as the initial state (lines 2 and 3). Then, the iterative procedure that constructs the exploration graph starts and continues until a certain number of states are added to the graph (lines 4 to 7). At each iteration, a new state $\boldsymbol{x}_{\mathbf{rand}}$ is randomly sampled in the search space (line 5), and it is added to the graph $\mathcal{G}$ with the Extend() procedure (line 6). The Extend() procedure is an essential step of the RRT$^{\#}$ algorithm because it extends the current graph by connecting $\boldsymbol{x}_{\mathbf{rand}}$ to the state with the minimum cost. Then, the Replan() procedure propagates all the updated costs on the graph, in order to update the graph accordingly (line 7). The Extend() and Replan() procedures are implemented exactly as in the original RRT$^{\#}$, for more details refer to [14]. After the graph is constructed, the algorithm defines the path for each robot position (lines 8 to 19). First, the algorithm verifies if the robot r with position $\boldsymbol{x}_{\mathbf{r}} \in \boldsymbol{X}_{\mathbf{r}}$ has the capabilities to perform the task k_{pr}. In case, the branch $\mathcal{T}$ connecting the task and robot positions

Algorithm 2 The GetCosts function implementing the multi-goal RRT$^{\#}$.

```
1  GetCosts(M_st, X, X_r, x(k_pr))
2      x_0 = x(k_pr);
3      G ← {x_0};
4      for i = 0 to N do
5          x_rand ← Sample();
6          G ← Extend(G, x_rand);
7          Replan(G);
8      foreach x_r ∈ X_r do
9          if M_st(r) = True then
10             T ← SpanningTree(G, x_r);
11             if T = ∅ then
12                 Costs ← {NaN};
13                 Times ← {NaN};
14             else
15                 Costs ← {c(T)};
16                 Times ← {t(T)};
17         else
18             Costs ← {NaN};
19             Times ← {NaN};
20     return Costs, Times
```

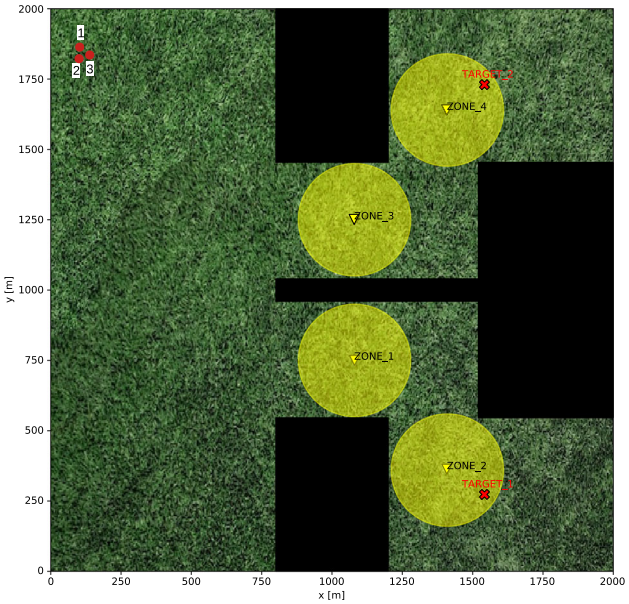

Fig. 5 The basic scenario evaluated in this work with the multi-robot system composed of 3 robots indicated with red circles in the upper left corner

is extracted from the graph (line 10), and the corresponding cost and time are included in the vector of costs and times, respectively. If a solution connecting the robot position $\boldsymbol{x}_{\mathbf{r}}$ and the task position $\boldsymbol{x}(k_{\text{pr}})$ does not exist, the cost and the time related to the robot-task combination are defined as NaN (Not a Number) (lines 12 and 13). A similar condition occurs if the robot is not suitable to perform the task (lines 18 and 19). Otherwise, when a solution connecting the robot position and the task position exists, the cost is defined considering the cost function used to compute the path, i.e. the path length in this paper. Instead, the time to reach the task position is estimated assuming that a robot moves at a constant speed. Then, the vectors Costs and Times are returned to the task allocation (line 20).

4 Results

In this section, the proposed task allocation and motion planning strategy is tested through simulations. The proposed strategy is implemented using the ROS (Robot Operating System) framework [34]. Specifically, the auction-based task allocation is implemented as a ROS node using Python, while the motion planner node is implemented using C++ and exploiting the OMPL (Open Motion Planning Library), an open-source library that contains several sampling-based motion planning algorithms [35].

In the following, the results have been split into four parts: first, we show how the proposed strategy is able to handle a basic scenario; second, the results related to dynamic scenarios with a human supervisor action are shown; third, we focus on the motion planner, showing the advantages of the proposed multi-goal strategy; finally, we show the advantages of adopting a synergetic combination of the auction-based allocation and the multi-goal RRT$^{\#}$ motion planning.

4.1 Basic Scenario

In this paragraph, we introduce the basic scenario considering a rescue operation as shown in Fig. 5. The main goal of the operation is to rescue two people with the same priority in

Table 1 Tasks with precedence for each sub-operation considering the basic scenario of Fig. 5

Sub-operation	Tasks
1	Fix(Zone_1) → Operate(Zone_1) → Fix(Zone_2) → Operate(Zone_2) → Rescue(Target_1)
2	Fix(Zone_3) → Operate(Zone_3) → Fix(Zone_4) → Operate(Zone_4) → Rescue(Target_2)

Table 2 Capabilities for each robot ($\boldsymbol{RC}$) considering the entire simulation time of the basic scenario

	p_1	p_2	p_3	p_4
r_1	x	x	x	x
r_2	x			x
r_3	x	x		x

the areas denoted as Target_1 and Target_2, therefore, in this example, the number s of targets is set to 2.

The black zones are obstacles ($X_{\mathbf{obs}}$), while yellow areas (Zone_1, Zone_2, Zone_3, and Zone_4) are zones to be adjusted to unlock the passage (Fix task), and, then, to be managed for example by extinguishing the fire (Operate task) to enable the navigation in that area by the robot in charge of rescue people (Rescue task).

In this example, the hierarchical structure shown in Fig. 1 is observed. Indeed, the final goal of the operation is to rescue two targets, i.e. two people with the same priority. Thus, the sub-operations are two and are composed of tasks with precedence. The tasks for each sub-operation are described in Table 1.

The first sub-operation, in Table 1, is to handle the Zone_1, Zone_2 and Target_1 sequentially. Instead, the second sub-operation is to handle the Zone_3, Zone_4 and Target_2 sequentially. Both the sub-operations have the same priorities and, then, can be performed simultaneously. Practically, the sub-operations force that Zone_1 and Zone_3 must be adjusted and managed before Zone_2 and Zone_4 and, lastly, people can be rescued in Target_1 and Target_2.

We assume that each robot can have at most 4 capabilities (i.e. $l = 4$). Table 2 shows the capabilities of each robot ($\boldsymbol{RC}$) belonging to the heterogeneous multi-robot system during the entire simulation time of the basic scenario. Thus, in this case, the capabilities of each robot remain unchanged throughout the simulation.

The capabilities p_1, p_2, p_3 and p_4 are particular features that empower a robot to accomplish a particular operation. In a practical rescue scenario, these capabilities aim to enhance the robot's effectiveness in saving lives and providing assistance during the emergency situations. For example, p_1 may refer to the ability to manipulate objects within the scenario, enabling to move obstacles and clearing the path required to reach the person in need of rescue. In our simulations, this capability is used in the "fix" task. Furthermore, p_2 may improve the robot performance during nighttime rescue operations. By incorporating special equipment to rescue the person (e.g. rescue ropes) and night vision, the robot is equipped to navigate and rescue people even in low-light conditions. On the other hand, p_3 may focus on daytime rescue operations. This capability provides the robot with equipment to rescue the person but does not include night vision, limiting its effectiveness to daylight hours. Lastly, p_4 may address the specific hazard of fires encountered during rescue operations. This capability, used in the "operate" task in our simulations, equips the robot with fire extinguishers.

Table 3 Capabilities for each task ($\boldsymbol{TC}$)

Task	p_1	p_2	p_3	p_4
Fix zone 1	x			
Operate zone 1				x
Fix zone 2	x			
Operate zone 2				x
Fix zone 3	x			
Operate zone 3				x
Fix zone 4	x			
Operate zone 4				x
Rescue target 1			x	
Rescue target 2		x		

Table 3 summarizes the capabilities needed for the execution of each task.

The auction-based task allocation, through the ongoing support of the motion planner, is able to successfully manage the basic scenario. Figure 6 shows the resulting plan that respects the precedence constraints between tasks, the heterogeneity of the team, and the prioritization between sub-operations. The time to reach the position for each task is estimated by the motion planner, considering the robot moving at constant speed.

4.2 Dynamic Scenario

The results of this section are obtained by evaluating the basic scenario of Fig. 5 but considering different perturbations at different instants. Thus, the auction-based task allocation is tested by simulating a dynamic scenario, and performing a re-allocation of the basic plan.

Specifically, results show how the system is able to handle both a sensor or a robot failure, and both the intervention of the human supervisor that decides to assign a task to a specific robot.

Figures 7 and 8 show the resulting plan after two different perturbations.

In the first condition, starting from the basic scenario, the re-allocation phase is triggered at the time instant of 500 s due to a failure of capability 4 on the second robot (see Table 4). Thus, the task allocator is called and the plan is re-allocated (see Fig. 7), thanks to the auction and the multi-goal motion planner.

In the second condition, starting from the basic scenario, the re-allocation phase is triggered at the time instant of 1500 s due to a total failure of the third robot. Thus, the whole

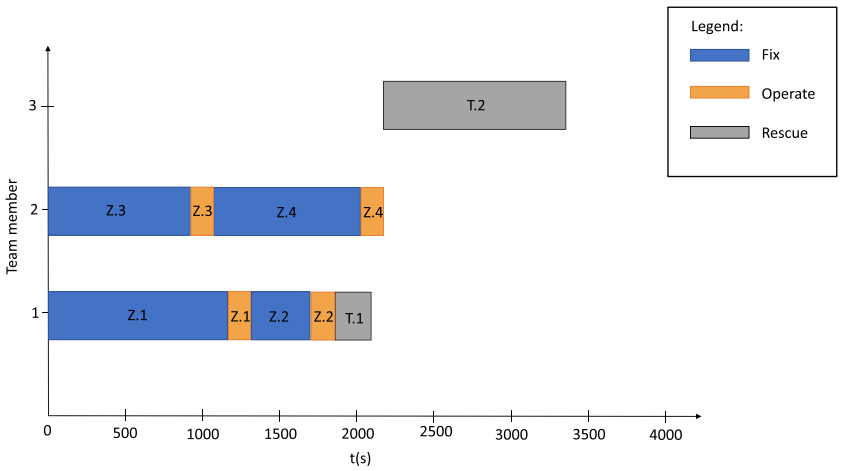

Fig. 6 Allocation of the basic scenario

system re-allocates the plan and the result is summarized in Fig. 8.

Another simulation is performed including an action of the human supervisor.

Starting from the basic scenario, Fig. 9 shows the plan after the action of the supervisor that forces the assignment of the task Fix(Zone_1) to the third robot. Here, the task allocator complies with this additional condition and allocates all other tasks accordingly, while respecting all the constraints we have detailed above.

4.3 Multi-goal Motion Planner

As previously defined, the motion planner plays a crucial role in the proposed strategy. Table 5 shows how the proposed motion planning improves the performance in terms of computational time without compromising the quality of the solution (i.e. the path length). The results of Table 5 compare the use of the standard RRT$^{\#}$ algorithm with the one with the multi-goal strategy proposed in this paper. Specifically, the values of Table 5 are the average ones of 20 executions of the scenario of Fig. 6.

The use of the standard RRT$^{\#}$ requires the computation of each path between a robot and task position. Hence, the motion planner is called several times in the scenario of Fig. 6. On the contrary, the use of the multi-goal RRT$^{\#}$ reduces the number of calls of the motion planner, since it computes simultaneously the paths between a task position and all the robot positions. As a consequence, the computational time is reduced. Moreover, Table 5 affirms that the quality of the solution in terms of path length does not change. The solution costs of the multi-goal RRT$^{\#}$ and original RRT$^{\#}$ are very similar. The small difference is due to the non-deterministic nature of the algorithm that never computes the same solution at each execution.

Another analysis is shown in Fig. 10, where the computational time between the multi-goal RRT$^{\#}$ and the original RRT$^{\#}$ is plotted as a function of the number of robots. Here, the path is computed between a fixed task position and several robots distributed in the scenario. Both multi-goal and original RRT$^{\#}$ generate an exploration graph of 5000 states

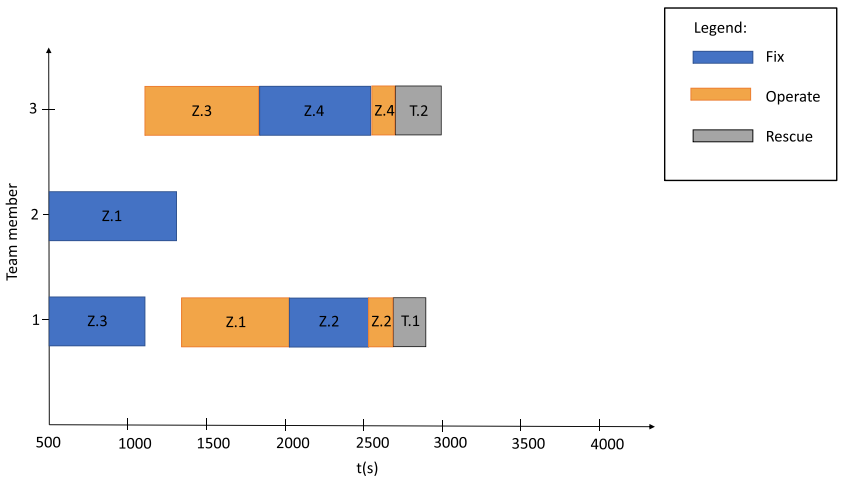

Fig. 7 Re-allocation starting from the basic scenario due to a failure of capability 4 of the second robot

Table 4 Capabilities for each robot after a sensor failure

	p_1	p_2	p_3	p_4
r_1	x	x	x	x
r_2	x			
r_3	x	x		x

to compute the path. As a result, the computational time required by the multi-goal RRT$^{\#}$ increases slower than the computational time of the original RRT$^{\#}$. This analysis confirms that the effectiveness of the proposed multi-goal RRT$^{\#}$ increases with the number of robots in the scenario.

4.4 Auction and Multi-goal Motion Planner

To demonstrate the effective synergy of the sequential-single item auction with the multi-goal motion planner, we conduct a computational time analysis for the basic scenario shown in Fig. 5. The analysis compares the computational time required by the sequential-single item auction implementing the original RRT$^{\#}$ with the one implementing the multi-goal RRT$^{\#}$, evaluating both the computation of the initial scheduling of Fig. 6 and the dynamic scheduling of Fig. 8. Simulations were executed on a laptop with Intel Core i5-10210U processor.

Regarding the initial scheduling, the proposed approach with the multi-goal RRT$^{\#}$ computes the solution of Fig. 6 in 2.27 seconds. Instead, the computational time required to compute a solution using the original RRT$^{\#}$ increases to 5.32 seconds. As previously discussed, this difference in the computational time is caused by the fact that the standard RRT$^{\#}$ is executed m (number of robots) times per each round of the auction. On the other hand, the multi-goal RRT$^{\#}$ is called only once per each round of the auction.

A similar trend is shown evaluating the dynamic scheduling of Fig. 8. The use of the multi-goal planner requires 1.16 seconds, while the use of standard RRT$^{\#}$ implies a computational time of 2.03 seconds. In this scenario, the computational time is lower because the task allocation problem involves only 2 robots and 6 tasks. This test highlights the benefits introduced by the proposed approach. Moreover, as also shown in Fig. 10, the benefits of our approach become evident as the number of robots increases.

Table 5 Comparison between the original and multi-goal RRT$^{\#}$ applied in the scenario of Fig. 5

	Computational Time [s]	Solution Cost [s]
original RRT$^{\#}$	1.716 (+45%)	356.609 (-0.1%)
multi-goal RRT$^{\#}$	1.182	356.953

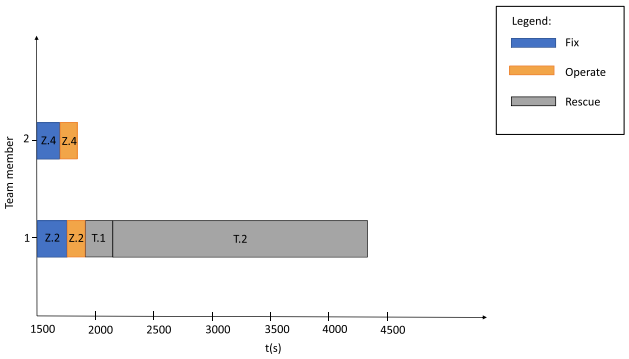

Fig. 8 Re-allocation due to a total failure of robot 3 starting from the basic scenario

5 Discussion and Conclusions

In this paper, a dynamic task allocation and a motion planning strategy for a team of heterogeneous robots are proposed by also including the interaction with a human supervisor. Specifically, the proposed solution consists of an auction-based task allocation, and a sampling-based motion planning based on the RRT$^{\#}$ algorithm and enhanced with a multi-goal approach.

We adopted a centralized architecture composed by a centralized task allocator and a centralized motion planner, since it offers three important advantages. First, the task allocator can directly interact with the motion planner, without avoiding delays caused by the communication with each agent of the HMRS. Second, the motion planner can parallelize path calculations. This logic could not have been adopted with the decentralized structure. Third, a centralized architecture is suitable for the interaction with a human supervisor. In this way, the supervisor has the possibility to intervene in the planning of the operation from a global point of view.

The proposed framework is tested in a simulation environment proving that our strategy is able to tackle a complex operation composed of different tasks in a dynamic scenario.

The proposed strategy is capable to handle the rescue operation of the basic scenario, as well as to handle perturbation events, e.g. sensor and robot failures. Results indicate that our approach can handle a multi-robot heterogeneous system in a dynamic scenario respecting precedence between tasks and priorities among sub-operations having computational efficiency as the main constraint since the system must be able to re-allocate on-the-go. This peculiarity is fulfilled by two features of our methodology: *(i)* the use of an auction-based task allocation with a human supervisor that is computationally efficient compared with MILP [16] and heuristic [32] approaches; *(ii)* the adoption of a motion planner with the multi-goal approach to take full advantage of the features of the sequential single-item auction.

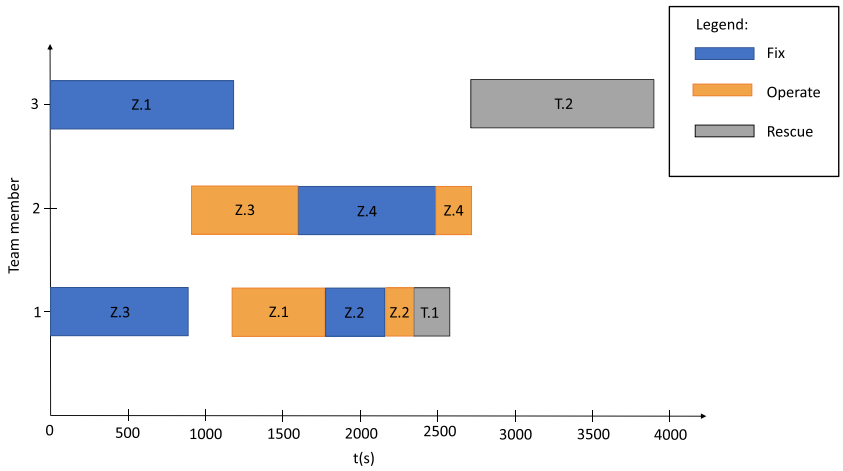

Fig. 9 Basic scenario with a supervisor decision. Indeed, the allocation of the task "Fix zone 1" has been assigned to the third robot by the supervisor and the remaining tasks have been allocated by the auction algorithm

The proposed multi-goal motion planner introduces several benefits to the overall system. A comparative analysis conducted in this study highlights the effectiveness of the proposed multi-goal motion planner in terms of computational time compared with the single-goal motion planner. This analysis proves the advantages introduced by the proposed method in terms of the scalability of the number of robots in the system and demonstrates the superiority of the sequential-single item auction when paired with the multi-goal RRT$^{\#}$.

Furthermore, unlike [29, 30], in this study we demonstrate that the multi-goal RRT$^{\#}$ guarantees an optimal path in the exploration graph. This is an essential feature because the quality of the solution of the auction-based task allocation strictly depends on the quality of the computed paths.

Moreover, the simulations with the interaction of the human supervisor led to promising results. The human supervisor is capable of constraining the plan by forcing the assignment to a specific robot or changing the capabilities required for tasks. Also in this scenario, the auction computes a valid plan respecting the constraints of the human supervisor. This is an important achievement since in the auction literature the human supervisor is almost never included [2, 19, 21].

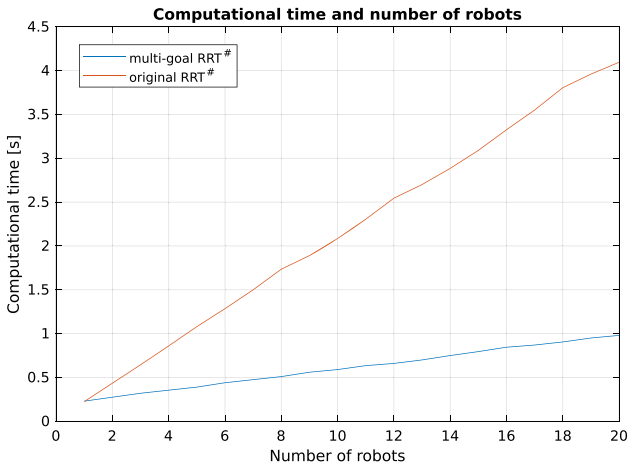

Fig. 10 Comparison of the computational time between the original and multi-goal RRT$^{\#}$ as a function of the number of robots

Despite the promising results, the proposed approach is not exempt from limitations. First, we do not account for the *stochasticity* of the duration of a task due to an event that was not predicted in the scenario. Second, although in the interest of the system's responsiveness, the solution obtained by the task allocation is *suboptimal,* since the adopted task allocation is based on a single-item auction. Our approach does not also take into account the *possibility of collaborations* between robots to perform tasks, nor does it contemplate temporal windows or deadlines for task completion. Moreover, the formulation of a low-level controller is required for the practical execution of the task on hardware.

The analysis of the limitations paves the way for possible improvements in the proposed strategy. For example, the *uncertainty* can be included in the estimation of the task execution time through the consideration of a specific probability distribution, along the lines of [36].

The suboptimality of the solution can be improved without affecting too much the computational efficiency by complementing the auction with heuristic approaches [37].

Collaborative tasks may be contemplated, similar to [2], where more than one agent collaborates in executing a task, together with time constraints in the execution times. The inclusion of all these aspects will affect the formulation and efficiency of the optimization problem –an aspect that notoriously lead to significant trade-offs.

In addition, a future implementation on hardware will call for the design of a low-level controller to materially execute the planned task once scheduled. Several well-established techniques have been proposed in the literature, such as in [38], where the authors present a control methodology for

a mobile robot in dynamic environments that contain both fixed and moving unforeseeable obstacles.

Furthermore, different operational aspects can involve different criteria for the design of the objective function, such as the minimization of the risk of the operation [39, 40], the travel distance, or the fuel consumption. The selection of these criteria could be operated automatically, or by the human operator according to his analysis of the operational scenario.

Finally, also unreliable communications between the robots and the task allocation unit should be considered and managed. This is a crucial issue in critical scenarios, such as in rescue operations in adverse weather conditions [19].

Author Contributions GG and SP conceived the research, designed the algorithms and produced the simulation results. All the authors contributed to the analysis of the results. GG and SP drafted a first version of the manuscript. AR supervised the research and produced a revised version of the manuscript. All the authors finally revised and agreed on the final version of the manuscript.

Funding Open access funding provided by Politecnico di Torino within the CRUI-CARE Agreement. This study was carried out within the MOST - Sustainable Mobility National Research Center and received funding from the European Union Next-Generation EU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) - MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.4 - D.D. 1033 17/06/2022, CN00000023). Moreover, it was carried out within the FAIR - Future Artificial Intelligence Research and received funding from the European Union Next-Generation EU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) - MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 - D.D. 1555 11/10/2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

Data Availability Not applicable, as no real data have been used to realize this work.

Code Availability The code will be made available upon request.

Declarations

Ethics Approval Not applicable. (This study does not involve human participants, their data or biological material).

Consent to Participate Not applicable (This article does not involve human participants).

Consent for Publication Not applicable (This article does not involve human participants).

Conflicts of interest The authors declare neither conflict of interest, nor competing interests.

References

1. Verma, J.K., Ranga, V.: Multi-robot coordination analysis, taxonomy, challenges and future scope. J. Intell. Robot. Syst. **102**(1), 1–36 (2021)
2. Wang, C., Wen, X., Niu, Y., Wu, L., Yin, D., Li, J.: Dynamic task allocation for heterogeneous manned-unmanned aerial vehicle teamwork. In: 2018 Chinese Automation Congress (CAC), pp. 3345–3349. IEEE (2018)
3. Mansouri, M., Pecora, F., Schüller, P.: Combining task and motion planning: challenges and guidelines. Front. Robot. AI, 133 (2021)
4. Suárez, S., Collins, J., López, B.: Improving rescue operations in disasters: approaches about task allocation and re-scheduling. In: The 24rd Annual Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG), pp. 66–75 (2005)
5. Albani, D., IJsselmuiden, J., Haken, R., Trianni, V.: Monitoring and mapping with robot swarms for agricultural applications. In: 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp. 1–6. IEEE (2017)
6. Jiang, Y., Hu, J., Lin, D.: Decision making of networked multiagent systems for interaction structures. IEEE Trans. Syst. Man Cybern. Syst. Hum. **41**(6), 1107–1121 (2011)
7. Ye, D., Zhang, M., Sutanto, D.: Self-adaptation-based dynamic coalition formation in a distributed agent network: a mechanism and a brief survey. IEEE Trans. Parallel Distrib. Syst. **24**(5), 1042–1051 (2013)
8. Murphy, R.R.: Human-robot interaction in rescue robotics. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **34**(2), 138–153 (2004)
9. Zanlongo, S.A., Dirksmeier, P., Long, P., Padir, T., Bobadilla, L.: Scheduling and path-planning for operator oversight of multiple robots. Robotics **10**(2), 57 (2021)
10. Gustavsson, M.E., Arnberg, F.K., Juth, N., von Schreeb, J.: Moral distress among disaster responders: what is it? Prehospital and Disaster Medicine **35**(2), 212–219 (2020)
11. Harbers, M., de Greeff, J., Kruijff-Korbayová, I., Neerincx, M.A., Hindriks, K.V.: Exploring the ethical landscape of robot-assisted search and rescue. In: A World with Robots: International Conference on Robot Ethics: ICRE 2015, pp. 93–107. Springer (2017)
12. Battistuzzi, L., Recchiuto, C.T., Sgorbissa, A.: Ethical concerns in rescue robotics: a scoping review. Ethics Inf. Technol. **23**(4), 863–875 (2021)
13. Dias, M.B., Zlot, R., Kalra, N., Stentz, A.: Market-based multirobot coordination: a survey and analysis. Proc. IEEE **94**(7), 1257–1270 (2006)
14. Arslan, O., Tsiotras, P.: Use of relaxation methods in sampling-based algorithms for optimal motion planning. In: 2013 IEEE International Conference on Robotics and Automation, pp. 2421–2428, IEEE (2013)
15. Gerkey, B.P., Matarić, M.J.: A formal analysis and taxonomy of task allocation in multi-robot systems. Int. J. Robot. Res. **23**(9), 939–954 (2004)
16. Bellingham, J., Tillerson, M., Richards, A., How, J.P.: Multi-task allocation and path planning for cooperating uavs. In: Cooperative Control: Models, Applications and Algorithms, pp. 23–41. Springer (2003)

17. Kurowski K., Stryk, O.V.: Online interaction of a human supervisor with multi-robot task allocation. In: Intelligent Autonomous Systems 13, pp. 965–978. Springer (2016)
18. Dai, W., Lu, H., Xiao, J., Zeng, Z., Zheng, Z.: Multi-robot dynamic task allocation for exploration and destruction. J. Intell. Robot. Syst. **98**(2), 455–479 (2020)
19. Otte, M., Kuhlman, M.J., Sofge, D.: Auctions for multi-robot task allocation in communication limited environments. Auton. Robot **44**(3), 547–584 (2020)
20. Nunes, E., Gini, M.: Multi-robot auctions for allocation of tasks with temporal constraints. In: Proceedings of the AAAI Conference on Artificial Intelligence, 29(1) (2015)
21. Moon, S., Oh, E., Shim, D.H.: An integral framework of task assignment and path planning for multiple unmanned aerial vehicles in dynamic environments. J. Intell. Robot. Syst. **70**(1), 303–313 (2013)
22. Liu, L., Shell, D.A.: Optimal market-based multi-robot task allocation via strategic pricing. In: Robotics: Science and Systems, vol.9, no.1, pp. 33–40 (2013)
23. Das, G.P., McGinnity, T.M., Coleman, S.A., Behera, L.: A distributed task allocation algorithm for a multi-robot system in healthcare facilities. J. Intell. Robot. Syst. **80**(1), 33–58 (2015)
24. Rizk, Y., Awad, M., Tunstel, E.W.: Cooperative heterogeneous multi-robot systems: a survey. ACM Computing Surveys (CSUR) **52**(2), 1–31 (2019)
25. Pippin, C., Christensen, H., Weiss, L.: Performance based task assignment in multi-robot patrolling. In: Proceedings of the 28th annual ACM symposium on applied computing, pp. 70–76 (2013)
26. Tan, K.C., Jung, M., Shyu, I., Wan, C., Dai, R.: Motion planning and task allocation for a jumping rover team. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 5278–5283. IEEE (2020)
27. Li, Z., Li, X., et al.: Research on model and algorithm of task allocation and path planning for multi-robot. Open J. Appl. Sci. **7**(10), 511 (2017)
28. Hussain, M., Kimiaghalam, B., Ahmedzadeh, A., Homaifar, A., Sayyarodsari, B.: Multi-robot scheduling using evolutionary algorithms. In: Proceedings of the 5th Biannual World Automation Congress, vol. 13, pp. 233–238. IEEE (2002)
29. Nanjanath, M., Gini, M.: Repeated auctions for robust task execution by a robot team. Robot Auton. Syst. **58**(7), 900–909 (2010)
30. Nanjanath, M., Gini, M.: Dynamic task allocation for robots via auctions. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, pp. 2781–2786, IEEE (2006)
31. LaValle, S.M.: Planning algorithms. Cambridge University Press, (2006)
32. Al-Hussaini, S., Gregory, J.M., Gupta, S.K.: Generating task reallocation suggestions to handle contingencies in human-supervised multi-robot missions. IEEE Trans. Autom. Sci, Eng (2023)
33. Öztürk, S., Kuzucuoğlu, A.E.: Optimal bid valuation using path finding for multi-robot task allocation. J. Intell. Manuf. **26**, 1049–1062 (2015)
34. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y., et al.: Ros: an open-source robot operating system. In: ICRA workshop on open source software, vol. 3, no. 3.2., p. 5. Kobe, Japan (2009)
35. Şucan, I.A., Moll, M., Kavraki, L.E.: The Open Motion Planning Library. IEEE Robot Autom. Mag. **19**(4), 72–82 (2012)
36. Shriyam, S., Gupta, S.K.: Task assignment and scheduling for mobile robot teams. In: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, vol. 51807, p. V05AT07A075. American Society of Mechanical Engineers (2018)
37. Cao, L., shun Tan, H., Peng, H., cong Pan, M.: Multiple uavs hierarchical dynamic task allocation based on pso-fsa and decentralized auction. In: 2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014), pp. 2368–2373 (2014)
38. Tanha, S., Dehkordi, S., Korayem, A.: Control a mobile robot in social environments by considering human as a moving obstacle. In: 2018 6th RSI International Conference on Robotics and Mechatronics (IcRoM), pp. 256–260. IEEE (2018)
39. Primatesta, S., Guglieri, G., Rizzo, A.: A risk-aware path planning strategy for uavs in urban environments. J. Intell. Robot. Syst. **95**, 629–643 (2019)
40. Primatesta, S., Cuomo, L.S., Guglieri, G., Rizzo, A.: An innovative algorithm to estimate risk optimum path for unmanned aerial vehicles in urban environments. Transp. Res. Procedia **35**, 44–53 (2018)

Giada Galati is a Ph.D. student in the Department of Electronics and Telecommunications at Politecnico di Torino, Italy. She received her M.Sc. in Mechatronics Engineering from Politecnico di Torino in 2019. In 2020 she started her Ph.D. with the Complex Systems Laboratory, directed by Prof. Alessandro Rizzo. Her field of research is human/robot interaction in multi-robot systems with applications to unmanned aerial vehicles and autonomous ground vehicles. Her research interests include also cooperative robotics.

Stefano Primatesta is an Assistant Professor with the Department of Mechanical and Aerospace Engineering. He received his Ph.D. in Computer and Control Engineering from Politecnico di Torino in 2019, and his B.S in Electronic Engineering and the M.S. in Mechatronic Engineering from Politecnico di Torino in 2011 and 2014, respectively. His field of research is the use of Remotely Piloted Aircraft Systems in urban environments including virtual modeling and multi-dimensional risk analysis. His research interests include also autonomous navigation and service robotics, with applications to unmanned aerial vehicles and unmanned ground vehicles.

Alessandro Rizzo received the Laurea degree (summa cum laude) in computer engineering and the Ph.D. degree in automation and electronics engineering from the University of Catania, Italy, in 1996 and 2000, respectively. In 1998, he worked as a EURATOM Research Fellow with JET Joint Undertaking, Abingdon, U.K., researching on sensor validation and fault diagnosis for nuclear fusion experiments. In 2000 and 2001, he worked as a Research Consultant at ST Microelectronics, Catania Site, Italy, and as an Industry Professor of robotics with the University of Messina, Italy. From 2002 to 2015, he was a tenured Assistant Professor with the Politecnico di Bari, Italy. Since 2012, he has been a Visiting Professor with the New York University Tandon School of Engineering, Brooklyn, NY, USA.

In November 2015, he joined Politecnico di Torino, where he is an Associate Professor in the Department of Electronics and Telecommunications and established the Complex Systems Laboratory. Dr. Rizzo is engaged in conducting and supervising research on complex networks and systems, modeling and control of nonlinear systems, and cooperative robotics. He is the author of two books, two international patents, and more than 200 papers on international journals and conference proceedings. He has been a recipient of the Award for the Best Application Paper at the IFAC world triennial conference in 2002 and of the Award for the Most Read Papers in Mathematics and Computers in Simulation (Elsevier) in 2009. He has also been a Distinguished Lecturer of the IEEE Nuclear and Plasma Science Society and the recipient of two Amazon Research Awards in robotics (2019 and 2021).