2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Optimal Task Allocation for Heterogeneous Multi-robot Teams with Battery Constraints

Álvaro Calvo[1] and Jesús Capitán[1]

***Abstract*— This paper presents a novel approach to optimal multi-robot task allocation in heterogeneous teams of robots. When robots have heterogeneous capabilities and there are diverse objectives and constraints to comply with, computing optimal plans can become especially hard. Moreover, we increase the problem complexity by: 1) considering battery-limited robots that need to schedule recharges; 2) tasks that can be decomposed into multiple fragments; and 3) multi-robot tasks that need to be executed by a coalition synchronously. We define a new problem for heterogeneous multi-robot task allocation and formulate it as a Mixed-Integer Linear Program that includes all the aforementioned features. Then we use an off-the-shelf solver to show the type of optimal solutions that our planner can produce and assess its performance in random scenarios. Our method, which is released as open-source code, represents a first step to formalize and analyze a complex problem that has not been solved in the state of the art.**

***Index Terms*— Multi-robot task allocation; Optimal planning and scheduling; Heterogeneous teams**

I. Introduction

The use of cooperative teams of heterogeneous robots is becoming a trend in many applications that can benefit from the combination of different sensing and/or locomotion robot capabilities, such as inspection [1], [2], precision agriculture [3]–[5], or fire fighting [6], [7]. For example, Unmanned Aerial Vehicles (UAVs) could be combined with ground robots [3], [4], as the former can access further places while the latter can load heavier equipment. The cooperation of heterogeneous UAVs is another option [6], [7], as they may provide different maneuverability (e.g., rotary vs. fixed-wing vehicles), or specific sensors and manipulation/delivery capabilities. In this context, optimal Multi-Robot Task Allocation (MRTA) becomes a hard problem, as there are typically multiple objectives to balance, as well as heterogeneous capabilities to accommodate and conflicting constraints to fulfill. In this work, we focus on multi-robot scenarios with two features particularly challenging: 1) battery constraints are quite limiting for UAVs, so recharging operations should be scheduled during operation; 2) tasks may require multiple robots to be executed, which implies an additional planning complexity and robot synchronization for task execution.

In the literature, there are numerous approaches to optimal MRTA. On the one hand, heuristic methods can achieve approximate solutions in a reasonable time; metaheuristic algorithms have been successfully applied to problems with heterogeneous robots and tasks with temporal and precedence constraints [4], [8], [9]. However, designing specific heuristics for the problem at hand is not straightforward in cases with multiple objectives and heterogeneous constraints [10]–[12]. On the other hand, there are exact approaches where the problem is formulated as a Mixed-Integer Linear Program (MILP) [5]. Most are based on variants of the well-known Vehicle Routing Problem (VRP), and many of them consider battery contraints for vehicles or heterogeneous capacities [2]. Recharging operations or inter-robot synchronization for multi-robot tasks are rarely addressed, though. For instance, Dorling et al. [9] presented a VRP for UAV delivery with recharging vehicles, and Li et al. [13] proposed an MILP formulation with multiple periods to model UAV recharges. Regarding multi-robot tasks that must be executed by several robots in a synchronized manner, algorithms have been proposed to compute optimal coalitions made up of robots with heterogeneous capacities [14], [15]. Moreover, Bredstrom et al. [16] proposed a VRP with temporal constraints for multi-robot synchronization, and Flushing et al. [17] an MILP for heterogeneous MRTA where *non-atomic* tasks are considered, i.e., tasks that can be executed incrementally over disjoint periods of time. We propose a related concept with fragmentable tasks that can be split into several segments, but we also add the possibility of having relays, which implies additional robot synchronization restrictions. Overall, among all existing MILP formulations of the state of the art, we believe that there is a gap of methods that combine the features addressed in this work.

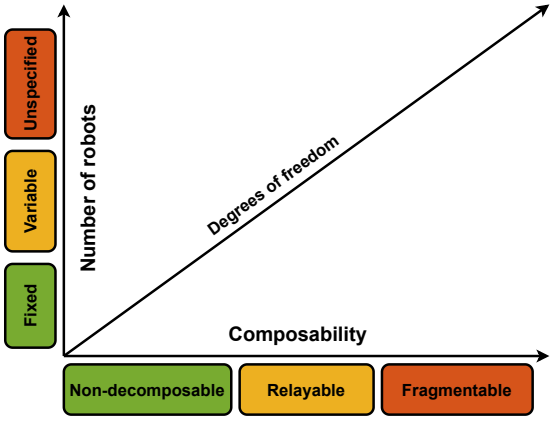


Fig. 1: Complexity of the problem according to the type of tasks in terms of number of robots and composability. The more degrees of freedom, the higher complexity.

[1]The authors are with the Multi-Robot Lab, University of Seville, Spain, [acalvom, jcapitan]@us.es.

This work is supported by project TED2021-129182B-I00, funded by MCIN/AEI/10.13039/501100011033 and the EU NextGenerationEU/PRTR, and the Spanish Government (PID2020-119027RB-I00).

Either multi-robot, fragmentable, and relayable tasks are not considered, or vehicle recharges are not modeled.

We advance the state of the art with a new approach for optimal MRTA for heterogeneous multi-robot teams with battery constraints and multi-robot tasks. We formulate the problem as a multi-objective MILP that accounts for a wide variety of possible options. First, we include recharge operations as additional tasks, which are then integrated into robots' schedules when needed. Second, we consider heterogeneous robots and a rich spectrum of task types (see Fig. 1), including: 1) tasks with fixed or variable number of required robots to be accomplished; 2) tasks that can be split into several fragments to be executed; and 3) tasks allowing relays between robots. Our main contributions are the following:

- We define a new MRTA problem for heterogeneous multi-robot teams (Section II), incorporating tasks of different types. To better analyze the complexity of the problem, we categorize it using the most well-known taxonomies in the literature.
- We derive a novel MILP formulation (Section III) to model the defined problem. The formulation proposes a multi-objective optimization, including recharging operations for the robots, fragmentable/relayable tasks, and multi-robot tasks with fixed or variable number of robots. In case of multi-robot tasks, the synchronization of the allocated robots is modeled by means of waiting times before triggering task execution.
- We use an off-the-shelf MILP solver to evaluate the efficacy and quality of our approach in a series of multi-UAV scenarios combining diverse restrictions (Section IV). Moreover, we provide our code as an open-source resource for the community.

Finally, we conclude and discuss future work in Section V. We believe that our work addresses a new MRTA problem of high interest for real multi-UAV applications, and that our proposed optimal solver will allow roboticists to better analyze the hardness of the problem and provide some hints to design adequate heuristics that resemble the optimal solutions and enable real-time task allocation.

II. Problem Description

We address a task planning problem with a team of co-operating heterogeneous robots. Robots are heterogeneous in the sense that they can provide different sensing/locomotion capabilities, and not all of them are suitable for each task. For instance, a robot with the ability to manipulate and transport items will be suitable for a delivery task, or a robot with a specific camera onboard could be required for a particular inspection task. Each robot has a limited operational time, but unlimited recharges are allowed (with an associated time cost) at fixed stations with known positions to reset the robot's battery level. Each task has a known spatial location and a predefined time duration; we do not consider specific time windows where tasks should start their execution or precedence constraints, but only a maximum completion time for each task (*deadline*), as a way to establish different priorities between tasks. The goal is to compute the optimal plan (minimizing the total mission completion time) for the team given all constraints. This means a set of ordered tasks that each robot has to execute. Our work is inspired by the use of heterogeneous teams of UAVs to execute long-endurance missions in outdoor settings. In these scenarios, the limited flight time of the UAVs is key, and this is why we consider explicitly recharging operations for extended autonomy. Now, let us describe the types of task that we consider in our problem:

- **Composability**: A task is *decomposable* if it can be split into several subtasks to be executed. We consider three types: *non-decomposable*, *fragmentable* and *relayable*. Non-decomposable tasks have to be executed entirely with the same set of robots from start to end without stop. Fragmentable tasks could be divided into a set of fragments (the original task duration would also be divided) that may be executed by different robots asynchronously, not necessarily with a given timeline or coordinately. On the contrary, relayable tasks can be divided into a sequence of fragments that must be executed in a continuous timeline without gaps in between. This division is done when a single robot or multi-robot coalition has no enough operational time to perform the task completely and there is a need for relays. Therefore, the different fragments of relayable tasks will be executed by different robots, since robots going to recharge are replaced by new ones.
- **Number of robots**: We consider three types of specification for the number of robots. First, tasks where the number of robots is *fixed*, either one or multiple. This is a hard constraint, so those tasks must be executed exactly by that number of robots. Second, tasks where the number of robots is *variable* as a soft constraint. This means that a number of robots is given as ideal for the task, but robot coalitions of different size are also allowed with a penalty. Third, tasks where the number of robots is *unspecified*. These are tasks to which no restriction or penalty is applied on the number of robots that have to execute the task, but the task duration will depend on the final number of robots allocated.

According to a well-known taxonomy of MRTA [18], we cover ST-MR-TA problems. This means that: our robots are *Single-Task* (ST), they can only perform one task at a time; our tasks are *Multi-Robot* (MR), they could require multiple robots to be executed; and we have a *Time-extended Assignment*, i.e., each robot is allocated several tasks that must be executed according to a given schedule. Nunes et al. [19] extended the previous taxonomy to differentiate between problems with *Task Windows* (TW) and *Synchronization and Precedence* (SP) constraints. Our problem falls within the TW category, we have no constraints on the starting times for the tasks but we do consider deadlines. Although we do not include precedence constraints between tasks, note that time synchronization is still imposed when multiple robots need to perform a task together or execute relays. Last, another well-

known taxonomy [20] has distinguished between different types of problems depending on the inter-task relationship. Our decomposable tasks fall into the category *Complex Dependencies* (CD) that they define. Given that we have MR tasks, there are inter-schedule dependencies, as plans for each robot cannot be computed independently. However, CD problems imply an additional complexity, as the optimal decomposition of tasks must be computed jointly with the task allocation. This is our case, where some tasks may be split for recharges and there is an additional degree of freedom to decide when to recharge.

III. MILP Formulation

In the following, we develop our mathematical formulation to solve the problem in Section II, cast as an MILP. Consider $\mathcal{R}$ the set of n heterogeneous robots; each robot $r \in \mathcal{R}$ has an initial position $p_{r,0} \in \mathbb{R}^3$, a traveling speed v_r, a maximum flight time F_r^{max} (battery autonomy), and an initial flight time consumed $F_{r,0}$. Consider $\mathcal{T}$ the set of m tasks to be executed by the team; each task $t \in \mathcal{T}$ has a spatial location p_t, an estimated execution time T_t^e, a deadline time to complete the task T_t^{max}, and a number of robots needed N_t [1]. Apart from the actual tasks, we include an additional task t_R so that the robots can recharge their batteries at one of the available base stations, defining $\tilde{\mathcal{T}} = \mathcal{T} \cup t_R$. Our formulation is agnostic to the method to select which is the *best* station to recharge at each moment, and we assume a fixed execution time $T_{t_R}^e$ to fully recharge the batteries.

The heterogeneous capabilities of robots are encoded through a set of binary variables $H_{r,t} \in \{0,1\}$, where $H_{r,t} = 1$ if robot r has the hardware required to execute the task t, and 0 otherwise. For each pair of tasks and robot, we define a displacement time T_{r,t_1,t_2}^d that is the estimated time to navigate robot r from the location of task t_1 to the location of t_2. We compute this navigation time using Euclidean distances between tasks (which may be available from a topological map) and the speed of each robot v_r. In order to consider the initial position of the robots, $t_1 \in \tilde{\mathcal{T}} \cup \mathcal{T}_0$ and $t_2 \in \tilde{\mathcal{T}}$, where $\mathcal{T}_0$ represents a set of n auxiliary fictitious tasks, each located in the initial position of each robot $p_{r,0}$.

Regarding the decision variables, we need to decide which tasks are allocated to each robot and in what order. For that, we introduce the concept of time slots: each robot has a task queue made up of a series of slots of variable duration (let $\mathcal{S}$ be the set of slots for each queue), where tasks can be allocated. The binary decision variables $x_{r,t,s}$ take value 1 if task t is assigned to slot s of robot r, and 0 otherwise. The duration of each slot will depend on the time required by the specific task placed in that slot, so slot durations will differ among robots, depending on the task assignment. In general, the sizes of the robot schedules (number of assigned tasks) can differ; however, for implementation purposes, all robot queues have the same size $|\mathcal{S}|$, and only the required number of slots gets *activated* for each robot through the variables $x_{r,t,s}$. Furthermore, since decomposable tasks can be split into several fragments, we also define the number of fragments in which each task is divided $n_t^f \in \mathbb{N}$, $n_t^f \in [1, N_f]$, where N_f is the maximum number of fragments into which any task can be divided [2]. Given the maximum flight time for robots, we can compute a bound for N_f by considering the worst case of the longest task and check the number of tours that would be needed (each tour implies a new fragment and a recharge) [3]. Note that each fragment is allocated to a different robot slot and its execution time is computed as T_t^e / n_t^f.

In multi-robot tasks, we enforce a time synchronization so that all the involved robots start the task simultaneously. A similar synchronization is needed when there is a relay in a relayable task. This is done by establishing a *waiting* time for each robot before starting, so that those arriving earlier wait for the others. Thus, we define $T_{r,s}^w$ as the time that robot r has to wait in slot s before starting the allocated task, to coordinate its execution with other robots. This value will depend on the arrival time of all the robots involved. We also define $T_{r,s}^d$ as the time required to move robot r to the location of the task allocated to slot s, starting at the location of its previous task assigned to slot $s-1$; $T_{r,s}^e$ as the time required to execute the task allocated to slot s; and $T_{r,s}^f$ as the time at which the task allocated to slot s finishes. Mathematically:

$$T_{r,s}^d = \sum_{t_2 \in \tilde{\mathcal{T}}} \Big(\sum_{t_1 \in \tilde{\mathcal{T}} \cup \mathcal{T}_0} T_{r,t_1,t_2}^d \cdot x_{r,t_1,s-1} \Big) \cdot x_{r,t_2,s} \tag{1a}$$

$$T_{r,s}^e = \sum_{t \in \tilde{\mathcal{T}}} (T_t^e / n_t^f \cdot x_{r,t,s}) \tag{1b}$$

$$T_{r,s}^f = T_{r,s-1}^f + T_{r,s}^d + T_{r,s}^w + T_{r,s}^e \tag{1c}$$

$$T_{r,0}^f = 0 \tag{1d}$$

$$\forall r \in \mathcal{R}, s \in \mathcal{S}$$

$F_{r,s}$ is the flight time accumulated by robot r at the end of slot s, which is computed recursively, being its initial value $F_{r,0}$. Equation (2a) takes into account that recharge tasks reset this consumed flight time to zero and that the waiting and execution times during recharge tasks do not consume flight time. Since robot batteries are limited, (2b) constraints the flight time consumed up to any slot to be no greater than the maximum flight time available, always leaving a minimum flight time F_r^{min} available for safety reasons. Robots should be able to fly back to a recharge station with that safety flight time.

$$F_{r,s} = F_{r,s-1} \cdot \bar{x}_{r,t_R,s-1} + T_{r,s}^d + (T_{r,s}^w + T_{r,s}^e) \cdot \bar{x}_{r,t_R,s} \tag{2a}$$

$$F_{r,s} \leq F_r^{max} - F_r^{min} \tag{2b}$$

$$\forall r \in \mathcal{R}, s \in \mathcal{S}$$

where $\bar{x}_{r,t,s} = 1 - x_{r,t,s}$.

[1] Note that this could be a hard or soft constraint depending on whether the task type regarding the number of robots is fixed or variable.

[2] This variable is forced to be 1 for recharge and non-decomposable tasks.

[3] A similar technique is used to compute a valid value for $|\mathcal{S}|$.

Additionally, we define some auxiliary variables for counting: $n_t \in \mathbb{N}$ counts the total number of times task t appears among all robot queues; $n_t^q \in \mathbb{N}$ counts the number of queues where task t appears in; and $n_t^r \in \mathbb{N}$ is the number of robots executing task t simultaneously. $n_{r,t}^q$ is a binary variable that takes the value 1 if task t appears in the queue of robot r. Formally:

$$n_t = \sum_{r \in \mathcal{R}} \sum_{s \in \mathcal{S}} x_{r,t,s} \tag{3a}$$

$$n_t^q = \sum_{r \in \mathcal{R}} n_{r,t}^q \tag{3b}$$

$$n_t^r \leq n_t^q \tag{3c}$$

$$n_t = n_t^r \cdot n_t^f \tag{3d}$$

$$\forall t \in \mathcal{T}$$

Apart from all the essential constraints explained so far, there are some additional constraints needed for our problem:

$$n_t^r \geq 1, \qquad \forall t \in \mathcal{T} \tag{4a}$$

$$\sum_{t \in \tilde{\mathcal{T}} \cup \mathcal{T}_0} x_{r,t,s} \leq 1, \qquad \forall r \in \mathcal{R}, s \in \mathcal{S} \tag{4b}$$

$$x_{r,t,s} \leq H_{r,t}, \quad \forall r \in \mathcal{R}, t \in \mathcal{T}, s \in \mathcal{S} \tag{4c}$$

$$x_{r,t_R,s-1} + x_{r,t_R,s} \leq 1, \qquad \forall r \in \mathcal{R}, s \in \mathcal{S} \tag{4d}$$

$$\sum_{t \in \tilde{\mathcal{T}}} x_{r,t,s} \leq \sum_{t \in \tilde{\mathcal{T}}} x_{r,t,s-1}, \quad \forall r \in \mathcal{R}, s \in \mathcal{S} \tag{4e}$$

First, we impose that all tasks are assigned to at least one robot (4a). We assume that the multi-robot team has the required hardware and size to accomplished all tasks in the scenario, and that the battery autonomy for the robots is enough to reach each task, execute it, and return to a base station. (4b) ensures that there is no more than one task per slot, (4c) imposes hardware compatibility of the robots assigned to a task, and (4d) avoids solutions with more than one consecutive recharge for the same robot. Finally, (4e) prevents the existence of free slots between tasks in a queue; tasks should occupy the lowest possible slots in the queue and empty slots should be after the last assigned task.

We propose a multi-objective cost function (5) with four terms to minimize: 1) the makespan, i.e., the time by which the last robot finishes its last task; 2) the delays for task completion with respect to their deadlines; 3) the waiting times for synchronization; and 4) the deviation of the number of robots assigned in multi-robot tasks, with respect to the ideal one.

$$f_1 = \frac{z}{\eta_1} \tag{5a}$$

$$f_2 = \frac{\sum_{r \in \mathcal{R}, s \in \mathcal{S}} \Delta T_{r,s}^{max}}{\eta_2} \tag{5b}$$

$$f_3 = \frac{\sum_{r \in \mathcal{R}, s \in \mathcal{S}} T_{r,s}^{w}}{\eta_3} \tag{5c}$$

$$f_4 = \frac{\sum_{t \in \mathcal{T}} U_t}{\eta_4}, \tag{5d}$$

where η_1, η_2, η_3, and η_4 are normalization constants so that all cost terms are in a similar scale. (5a) introduces an auxiliary variable z to encode the makespan, which can be done by adding the constraint $z \geq T_{r,|\mathcal{S}|}^f, \forall r \in \mathcal{R}$, i.e., z must be greater than the completion time for each robot's queue (1c). η_1 is computed by considering a worst-case scenario in which a single robot executes all tasks and recharging when needed. In (5b), $\Delta T_{r,s}^{max}$ are slack variables that represent the delay of robot r in completing the task assigned to slot s. We impose:

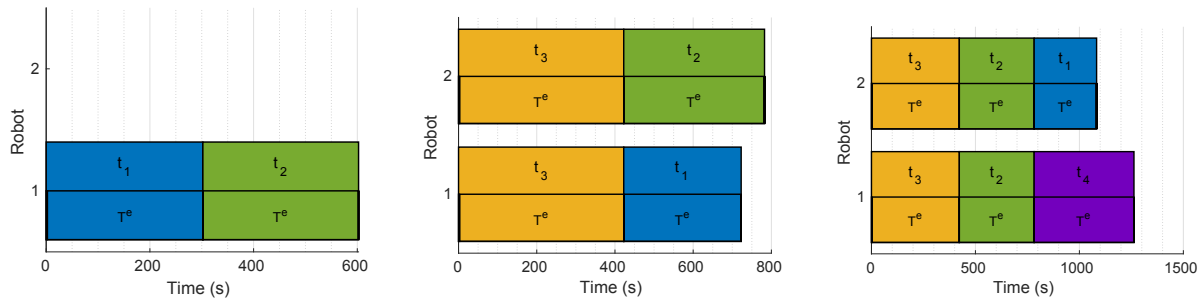
$$\begin{aligned} &\Delta T_{r,s}^{max} \geq \sum_{t \in \mathcal{T}} x_{r,t,s} \cdot (T_{r,s}^f - T_t^{max}), \\ &\Delta T_{r,s}^{max} \geq 0, \quad \forall r \in \mathcal{R}, s \in \mathcal{S} \end{aligned} \tag{6}$$

Note that, as it does not make sense to consider deadlines for recharges or unassigned slots, (6) reduces to $\Delta T_{r,s}^{max} \geq 0$ when the task assigned to slot s does not belong to $\mathcal{T}$. η_2 is computed as the maximum deadline $T_t^{max}, \forall t \in \mathcal{T}$. (5c) is the normalized overall waiting time, where η_3 is calculated as the maximum battery time autonomy $\max\{F_r^{max}\}, \forall r \in \mathcal{R}$. Last, recall that our multi-robot tasks may have a *variable* number of robots, so (5d) penalizes tasks with an allocated number of robots that differs from their ideal N_t. This deviation in the number of assigned robots is defined as:

$$V_t = N_t - n_t^r, \quad \forall t \in \mathcal{T} \textit{ if } N_t > 0. \tag{7}$$

Tasks with a variable number of robots may get fewer or more robots assigned than their specified N_t, so V_t can be positive or negative, and we want to minimize its absolute value. For that, we minimize the auxiliary variable U_t, limited to $U_t \geq -V_t$ and $U_t \geq V_t$. Therefore, $U_t \in [|V_t|, +\infty)$. For tasks with an *unspecified* number of robots, this penalty does not apply; by convention, we set $N_t = 0$ for those tasks, which is why (7) only applies when this parameter is greater than zero. For tasks with a *fixed* number of robots, an extra constraint is added to force V_t to be zero, making n_t^r and N_t coincide. η_4 is computed by adding the maximum deviation for all tasks, where this maximum deviation is $\max\{n - N_t, N_t - 1\}$; i.e., the worst case of the two extremes: allocating a single robot or all robots to the task.

Finally, it is important to remark that some of the described equations contain non-linear elements, such as decision variables that multiply each other (see, for instance, (1)). In order to keep the formulation as an MILP, we circumvent this issue with linearization techniques by using additional auxiliary variables [21], [22]. Basically, non-linear terms are replaced by new variables and constraints so that the values of the original variables keep unchanged. Moreover, there are more secondary constraints in our complete MILP, mainly related to task synchronization. Those are needed to coordinate temporally the execution of different instances of the same task (in the case of multi-robot tasks) or different fragments of the same task (in the case of relays). All these implementation details related to linearization and synchronization are not shown here due to space restrictions but can be consulted in our code online (see Section IV).



(a) Hardware compatibility: only one robot is compatible with the tasks.

(b) Multi-robot task with variable number of robots.

(c) Multi-robot task with variable number of robots.

(d) Recharging tasks allowed.

(e) Recharging tasks not allowed.

(f) Fragmentable multi-robot task.

(g) Task composability test: relayable task.

(h) Task composability test: non-decomposable task.

Fig. 2: Solution plans for example scenarios illustrating the capabilities of the method.

IV. Experimental Results

This section shows simulation results with two objectives: (i) illustrate the capabilities of our method to cope with different types of task; and (ii) assess its performance in terms of solution quality. First, we propose simple scenarios including the features described in Section II and demonstrate how the optimal solutions obtained by our method can leverage the inclusion of robot recharges, task fragmentation, and multi-robot tasks with a flexible number of assigned robots. Second, we test the method through a set of random scenarios to analyze different performance metrics.

We use the well-known off-the-shelf solver Gurobi [4] to compute optimal solutions. In particular, we implemented our MILP described in Section III using the MATLAB interface for Gurobi. The code with the full MILP implementation and the simulations in this section can be accessed online [5].

[4] https://www.gurobi.com.

[5] https://github.com/multirobot-use/task_planner

A. Illustrative examples

We designed manually a set of simple scenarios to show the capability of our planner to deal with different situations in a separate manner, which gives a hint of its potential to solve more complex missions that combine all those features. Fig. 2 depicts the plans computed for each robot in all scenarios. Different colors are used to represent each task in each robot's timeline, with recharge tasks colored in orange. For the sake of simplicity, we only report here the values of the parameters that are relevant to understand each scenario and omit the rest.

Fig. 2a shows a hardware compatibility test, where robot 2 is unable to perform any of the two existing tasks while robot 1 can perform both. It can be seen that the solver accounts for capability constraints and assigns all tasks to robot 1. In the remaining examples, we remove the capability constraints by setting $H_{r,t} = 1, \forall r \in \mathcal{R}, t \in \mathcal{T}$. Fig. 2b and 2c show an example to illustrate the advantage of using a variable number of robots for multi-robot tasks that allow it. In

MILP VARIATION	SUCCESS RATE (%)	MAKESPAN (MIN)	TOTAL TRAVEL TIME (MIN)	TOTAL WAITING TIME (MIN)	SOLUTIONS WITH RECHARGES (%)
Complete	92	52.19 ± 22.51	56.21 ± 40.55	6.45 ± 10.87	83
No-Recharges	16	17.69 ± 0.84	0.85 ± 0.32	0.00 ± 0.00	0
Incomplete	16	39.65 ± 1.36	40.60 ± 9.72	0.15 ± 0.11	100

TABLE I: Performance results for 3 variants of the planner. Average values and standard deviations are shown.

Fig. 2b, there are 3 tasks with $N_1 = 1$, $N_2 = 2$ and $N_3 = 2$; tasks t_1 and t_3 require a fixed number of robots, whereas task t_2 can be variable. In Fig. 2c, there is an additional task t_4 with fixed $N_4 = 1$. In the first case, it can be seen how t_2, despite the penalty of not matching N_2, is executed with a single robot, because the total makespan of the mission gets reduced that way. However, in the second case, given the extra task, no time is saved by doing that, so t_2 is executed by the two robots as a preferred option. To check the advantage of including recharges, Fig. 2d and 2e depict solutions for the same scenario with 3 single-robot tasks considering and not considering recharges, respectively. In that scenario, robot 2 has only enough initial flight time to execute t_1. Without recharges, robot 1 must execute the other two longer tasks (the makespan is 40 minutes); when recharges are allowed, robot 2 can start recharging to execute one of the long tasks later, hence reducing the total makespan of the team to 25 minutes. Next, another scenario with a multi-robot fragmentable task ($N_1 = 2$) is shown in Fig. 2f. Both robots execute fragments of the task together, interleaving recharging operations with their corresponding synchronizations. Finally, we design a scenario with two single-robot tasks that demonstrates the usefulness of introducing relays in tasks that are decomposable. Fig. 2g shows the optimal solution if t_2 is treated as a relayable task; robot 1 is relayed by robot 2 while recharging and, once recharged, it resumes with the task, with a final makespan of 20 minutes. Note that in both relays, robot 2 and robot 1 plan some waiting time for synchronization. Fig. 2h shows the optimal solution if the same task is treated as non-decomposable, it can be seen that the makespan increases (25 minutes).

B. Performance evaluation

We evaluate the quality of the method solutions in a set of scenarios randomly generated. As performance metrics, we measure for each scenario the makespan, the total travel time for all robots, the total waiting time, and the total number of recharges. The success rate indicates the percentage of scenarios where a solution is found. We generated 50 scenarios with 2 robots and 2 tasks as follows [6]. All robots have $F_r^{max} = 20\ min$, $F_r^{min} = 1\ min$, $F_{r,0} = 0\ min$, $v_r = 5\ m/s$, and an initial position uniformly sampled from a 3D space of size $200\ m \times 300\ m \times 5\ m$. A single recharge station is always located in the middle of the scenario, and the execution time for recharges is $5\ min$. There are no hardware constraints, and all robots can perform all tasks. Each task has a time of execution T_t^e uniformly sampled from the range $[15, 60]\ min$ and the same deadline $T_t^{max} = 60\ min$. Task types depend on several parameters that are randomly chosen. Afterwards, we remove scenarios that are unsolvable a priori, e.g., we ensure that all non-decomposable tasks are short enough to be covered with the maximum flight time. As a result, tasks are randomly categorized as relayable, fragmentable, and non-decomposable, with resulting probabilities of 0.25, 0.625, and 0.125, respectively. Besides that, they can have an unspecified number of robots or a variable $N_t = 2$, both cases with equal probability.

As it is difficult to find alternative state-of-the-art approaches that address our complex optimization problem in an exact manner, we compare three variants of our planner: the *Complete* version solves the MILP with all its features activated; the *No-Recharges* version does not allow robots to recharge; and the *Incomplete* version allows recharges but not task relays, fragmentation, or variable N_t. Table I depicts the results. The three methods show a meaningful difference in success rate. The Complete method can solve almost all the scenarios, while No-Recharges and Incomplete methods fail to solve scenarios that are unfeasible without recharging, task fragmentation and/or variable N_t. Although the makespan and travel time are higher for the Complete approach, this is because the metrics are computed over the solvable scenarios; the No-Recharges and Incomplete variants can only solve scenarios with shorter tasks, that is, with lower makespan and travel time. We have also noticed that other terms of the optimization cost function worsen, so the total solution costs improve with the Complete variant. Moreover, the Complete method finds more efficient solutions, reducing the percentage of scenarios where recharging is needed. The higher waiting time in the Complete results indicate more synchronizations and relays to achieve those efficient solutions.

V. CONCLUSIONS

We have proposed an MILP formulation for heterogeneous MRTA with battery recharges and tasks that can be decomposed and executed by multiple robots in a synchronized manner. Our results demonstrate that our method can cope with a diverse set of features, including task fragmentation and robot relays, which could yield complex scenarios of interest for a wide spectrum of multi-robot applications. However, the solver does not scale well, and our future work will concentrate on heuristic methods that can find solutions more efficiently. Still, this work represents a first step to formalize a new MRTA problem, and the presented optimal planner can serve to better analyze the complexity of the problem and to provide solutions that help us designing adequate heuristics that enable real-time task allocation.

[6]Scenarios of higher dimensions were not tested due to computational issues.

REFERENCES

[1] A. Calvo, G. Silano, and J. Capitan, "Mission Planning and Execution in Heterogeneous Teams of Aerial Robots supporting Power Line Inspection Operations," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, Dubrovnik, Croatia, 2022, pp. 1644–1649.

[2] S. Agarwal and S. Akella, "Line coverage with multiple robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3248–3254.

[3] P. Tokekar, J. V. Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic UAV and UGV system for precision agriculture," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1498–1511, 2016.

[4] B. A. Ferreira, T. Petrović, M. Orsag, J. R. M. de Dios, and S. Bogdan, "Distributed allocation and scheduling of tasks with cross-schedule dependencies for heterogeneous multi-robot teams," *Arxiv Pre-prints*, 2021. [Online]. Available: http://arxiv.org/abs/2109.03089

[5] K. Leahy, Z. Serlin, C.-I. Vasile, A. Schoer, A. M. Jones, R. Tron, and C. Belta, "Scalable and Robust Algorithms for Task-Based Coordination From High-Level Specifications (ScRATCHeS)," *IEEE Transactions on Robotics*, pp. 1–20, 2021.

[6] F. Real, A. R. Castaño, A. Torres-González, J. Capitán, P. J. Sánchez-Cuevas, M. J. Fernández, H. Romero, and A. Ollero, "Autonomous fire-fighting with heterogeneous team of unmanned aerial vehicles," *Field Robotics*, vol. 1, pp. 158–185, 2021.

[7] E. Seraj, L. Chen, and M. C. Gombolay, "A hierarchical coordination framework for joint perception-action tasks in composite robot teams," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 139–158, 2022.

[8] B. Miloradovic, B. Curuklu, M. Ekstrom, and A. V. Papadopoulos, "GMP: A Genetic Mission Planner for Heterogeneous Multirobot System Applications," *IEEE Transactions on Cybernetics*, vol. 52, no. 10, pp. 10 627–10 638, 2022.

[9] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 1, pp. 70–85, 2017.

[10] S. Al-Hussaini, J. M. Gregory, and S. K. Gupta, "Generating task reallocation suggestions to handle contingencies in human-supervised multi-robot missions," *IEEE Transactions on Automation Science and Engineering*, pp. 1–15, 2023.

[11] G. Neville, S. Chernova, and H. Ravichandar, "D-ITAGS: A Dynamic Interleaved Approach to Resilient Task Allocation, Scheduling, and Motion Planning," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 1037–1044, 2023.

[12] 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.

[13] M. Li, L. Zhen, S. Wang, W. Lv, and X. Qu, "Unmanned aerial vehicle scheduling problem for traffic monitoring," *Computers & Industrial Engineering*, vol. 122, pp. 15–23, 2018.

[14] S. D. Ramchurn, M. Polukarov, A. Farinelli, N. Jennings, and C. Trong, "Coalition formation with spatial and temporal constraints," in *International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2010, pp. 1181–1188.

[15] W. Gosrich, S. Mayya, S. Narayan, M. Malencia, S. Agarwal, and V. Kumar, "Multi-robot coordination and cooperation with task precedence relationships," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 5800–5806.

[16] D. Bredström and M. Rönnqvist, "Combined vehicle routing and scheduling with temporal precedence and synchronization constraints," *European Journal of Operational Research*, vol. 191, no. 1, pp. 19–31, 2008.

[17] E. Feo Flushing, L. M. Gambardella, and G. A. Di Caro, "A mathematical programming approach to collaborative missions with heterogeneous teams," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 396–403.

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

[19] E. Nunes, M. Manner, H. Mitiche, and M. Gini, "A taxonomy for task allocation problems with temporal and ordering constraints," *Robotics and Autonomous Systems*, vol. 90, pp. 55–70, 2017.

[20] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[21] A. Anirudh Sabnis, R. K. Sitaraman, and D. Towsley, "OCCAM: An Optimization Based Approach to Network Inference," *SIGMETRICS Performance Evaluation Review*, vol. 46, no. 2, p. 36–38, 2019.

[22] W. Tan and B. Khoshnevis, "A linearized polynomial mixed integer programming model for the integration of process planning and scheduling," *Journal of Intelligent Manufacturing*, vol. 15, no. 5, pp. 593–605, 2004.