A Formal Definition of the Multi-Robot Multi-Task Time-extended Assignment Problem Configuration

Branko Miloradović[1], Alessandro V. Papadopoulos[1]

***Abstract*— Multi-Robot Systems (MRSs) play a crucial role in several fields, including industrial automation, precision agriculture, and urban search and rescue, by enhancing efficiency and operational capabilities. One of the main challenges in these systems is the efficient allocation of tasks, known as Multi-Robot Task Allocation (MRTA). This paper focuses on a particularly complex configuration of MRTA that involves Multi-Task (MT) Robots capable of performing multiple tasks simultaneously, Multi-Robot (MR) Tasks that require coordinated efforts from several robots, and Time-extended Assignments (TA) that demand extended duration scheduling. Despite notable advancements in this area, the MT-MR-TA configuration remains underexplored. Existing research often fails to address the specific challenges associated with coordinating and scheduling these complex tasks. This paper aims to fill this gap by introducing new computational models based on Integer Linear Programming (ILP) and Constraint Programming (CP). These models are purposefully designed to formalize and tackle the intricate dynamics of MT-MR-TA, offering a structured approach to solve this multidimensional optimization problem. We rigorously evaluate these models for their effectiveness using advanced, general-purpose solvers across various instances, with a focus on model scalability, solver efficiency, and overall solution quality.**

I. Introduction

Multi-Robot Systems (MRSs) have become essential in advancing various sectors, including industrial automation, precision agriculture, and urban search and rescue. By coordinating multiple robots, these systems can tackle tasks that single robots cannot manage, which enhances both operational efficiency and effectiveness. As the complexity and quantity of tasks increase, effective task allocation among robots, known as Multi-Robot Task Allocation (MRTA), becomes critical [4]. MRTA aims to optimize the distribution of tasks among robots to maximize efficiency and resource utilization while adhering to the operational constraints of each mission. Gerkey and Matarić [8] provide foundational analysis and taxonomy of MRTA, offering insights into its various forms and challenges. This complexity has grown as robots have become capable of more sophisticated and diverse tasks. This paper addresses a particularly challenging aspect of MRTA: the Multi-Task (MT) Multi-Robot (MR) Time-extended Assignment (TA) Problem Configuration with Cross-Schedule Dependencies and Precedence Constraints.

Research on MRTA has been extensive, leading to the development of various taxonomies and frameworks to categorize its challenges and configurations. Korsah et al. [11] provide a comprehensive taxonomy that serves as a framework for understanding the complex nature of MRTA configurations, with a focus on the interplay between robot capabilities and task requirements. Nunes et al. [18] explore the temporal and ordering constraints inherent in task allocation, which are critical for planning and executing time-sensitive tasks in multi-robot operations. Despite significant advances in MRTA research, the specific challenges of scheduling tasks that require concurrent action by multiple robots over extended periods have not been thoroughly examined. Existing research often overlooks the integrated challenge of MT-MR-TA, which calls for innovative problem-solving approaches. This paper aims to fill this gap by focusing on the MT-MR-TA configuration within MRTA.

Our contribution is threefold: (1) we introduce novel computational models: Integer Linear Programming (ILP) and Constraint Programming (CP). These models are specifically designed to address the unique challenges of the MT-MR-TA configuration, providing a formal framework for defining and solving these complex multi-dimensional optimization problems; (2) we perform a rigorous evaluation of these models using advanced, general-purpose solvers to assess their scalability, efficiency, and the quality of the solutions they produce. This evaluation provides new insights into the applicability and performance of ILP and CP in complex MRTA scenarios; and (3) we enhance the current theoretical and practical understanding of MRTA by detailing the implementation and validation of our models in a variety of real-world settings, thereby contributing to both academic research and practical applications in the field. By addressing these key aspects, our study not only enhances the theoretical framework of MRTA but also provides practical solutions to significant gaps in existing research. We expect our findings to facilitate more sophisticated implementations of MRTA in practice, paving the way for future advancements in the field of multi-robot systems [1], [7], [2].

II. Background and Motivation

In the pioneering MRTA taxonomy proposed by Gerkey and Matarić [8], three key dimensions were identified, leading to a classification of 8 distinct problem configurations. This classification system includes (i) *Single-Task (ST)* versus *Multi-Task (MT)* robots, distinguishing between scenarios where robots can handle only one task concurrently versus those where robots manage multiple tasks simultaneously; (ii) *Single-Robot (SR)* versus *Multi-Robot (MR)* tasks, discerning tasks that can be accomplished by either a single

This work is funded by The Knowledge Foundation (KKS), MARC project No. #20240011

[1]Division of Intelligent Future Technologies, Mälardalen University, Västerås, Sweden. Email: {branko.miloradovic, alessandro.papadopoulos}@mdu.se

robot or require multiple robots for completion; (iii) *Instantaneous Assignment (IA)* versus *Time-extended Assignment (TA)*, contrasting situations where robots have information about only the next task versus those where they are provided with a complete schedule of tasks. Within this taxonomy, the MT-MR-TA problem configuration serves as a subset of the problem investigated in this study. In addition, we augment this problem configuration by incorporating additional dimensions from other taxonomies. Nevertheless, it is essential to highlight that our problem can be reduced to the MT-MR-TA problem configuration. This simplification forms a reasonable basis for explaining the problem addressed in this paper. Moreover, the interrelatedness of the tasks has not been addressed in [8], which makes the definition of MT or MR configurations incomplete.

The subsequent advancement in the MRTA taxonomy, introduced by Korsah *et al.* [11], presented the framework *iTax*. This framework addressed task interdependencies, a crucial aspect absent in the original MRTA taxonomy. According to *iTax*, when tasks demonstrate interrelations within the schedule of the same robot, the problem falls in the Intra-schedule Dependencies (ID) group, resembling ordering constraints. Conversely, if tasks scheduled for execution by different robots display interrelations, such as Precedence Constraints (PCs), it indicates dependencies between those schedules, termed Cross-schedule Dependencies (XD). Although *iTax* covers additional task dependencies, these aspects lie beyond the scope of this paper. This work also offers a survey encompassing various problem configurations. Notably, the MT-MR-TA configuration lacked a formal problem definition. Moreover, in tables summarizing the literature from surveyed papers on previously addressed problem configurations, the MT-MR-TA entry remained empty.

Following the taxonomy extension by Nunes *et al.* [18], we add the last missing piece to describe the problem configuration addressed in this paper. In [18], two new dimensions were introduced: Synchronization and Precedence (SP) and Time Windows (TW). SP signifies either synchronization between two tasks (e.g., two tasks have to start at the same time) or precedence constraints (e.g., one task precedes the other). Conversely, TW represents a constraint mandating a specific task to occur within a predefined time slot. This study concentrates primarily on exploring the SP dimension. Additionally, [18] extensively reviewed the literature encompassing various other MRTA problem types. Regarding the [XD]:MT-MR-TA problem configuration, their investigation highlighted that this particular aspect of the MRTA problem domain remained unexplored. As SP helps to extend the definition of XD dependencies, we can finally formulate our problem configuration as **[XD]:MT-MR-TA-(SP)**. A visual representation of this problem configuration is given in Fig. 1. Even in the most recent extension of the MRTA taxonomy named TAMER [16], there has yet to be further exploration of the MT-MR-TA segment; instead, it remains unchanged from previous MRTA taxonomies. Similarly, in a recent survey paper on MRTA [5], the authors could not find any paper in the literature addressing the MR-MT-TA problem configuration.

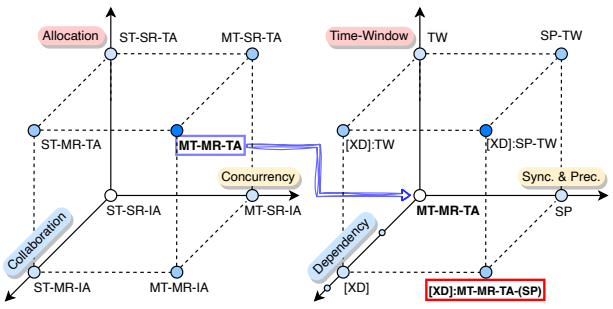

Fig. 1: [XD]:MT-MR-TA-(SP) problem configuration defined through MRTA taxonomies.

Upon examining the surveys conducted in the most influential MRTA taxonomies, it is apparent that MT-MR-TA has received scant attention as a viable research direction, both theoretically and practically. This neglect persists to this day. However, the underlying reasons have yet to be addressed. We discuss two primary factors contributing to this oversight. Firstly, the perceived issues revolve around the cost and complexity of implementing such systems, making them seemingly impractical for real-world applications. However, there has been a notable increase in affordable, sophisticated robotic systems on the market, hinting at a potential future where widely accessible hardware can handle the demanding tasks posed by the MT-MR-TA problem. Secondly, the complexity of this problem configuration is a considerable deterrent. It stands as one of the most intricate problem configurations within MRTA taxonomies. No known solver is specifically tailored for this optimization problem [5]. Moreover, the inherent complexity might confine the acquisition of feasible, satisfactory solutions solely to smaller instances of the problem. The only related work we could find on MT-MR-TA problem configuration [20], [19], [6] is not truly MT-MR-TA, mostly due to the lack of MT part of the problem, as defined in the original MRTA taxonomy. We expanded our investigation into problems similar to this one beyond the area of robotics, specifically exploring the domain of Operations Research (OR). While most dimensions of the Multi-Robot Task Allocation (MRTA) problem can be mapped to existing OR problems, the [XD]:MT-MR-TA-(SP) dimension appears to lack a direct counterpart in OR. The closest match we found is the Multi-Skilled Resource-Constrained Project Scheduling Problem (MSRCPSP) [10], [22], where tasks correspond to activities, robots to resources, and skills to task/robot capabilities. What MSRCPSP is lacking is the ability to handle parallelism on the robot's level, which ultimately makes it analogous to MRTA's ST-MR-TA dimension. In addition, in MSRCPSP there is no travel time between activities.

We aim to advance the theoretical understanding of the problem. Our effort intends to spotlight the [XD]:MT-MR-TA-(SP) (we will refer to it simply as MT-MR-TA) configuration by employing various problem models and utilizing publicly available solvers to tackle different problem instances. This will help us gain insight into the problem's difficulty

and the capability of currently available general-purpose solvers to handle this complex problem. Hopefully, this work will inspire researchers to develop problem-specific heuristic solvers to tackle larger problem instances successfully.

III. Task Types

We separate tasks into **virtual** and **physical**. Virtual tasks lack spatial constraints during execution, allowing them to be carried out in any physical location and leveraging the computing platform's parallelism capabilities. Examples of virtual tasks encompass activities such as communication, sending proprioceptive data, data analysis, etc. These tasks can be executed independently or concurrently with other tasks. In contrast, physical tasks are confined to specific locations for execution and cannot be performed during transitions between tasks. From a modeling perspective, if two or more physical tasks can run in parallel at the same location, they should be represented as a single task associated with that location. The duration of this task can be determined either by taking the maximum duration of the individual tasks (if they can all run in parallel) or by solving a local scheduling problem to minimize the makespan of the physical tasks. For instance, a robot with two arms manipulating two different objects can be seen as executing two separate, independent tasks in parallel. Therefore, multiple physical tasks can be modeled as one monolithic task. These tasks involve physical interactions such as object manipulation, environmental sensing (e.g., scanning the seabed, capturing images of specific objects), or utilizing tools (e.g., drilling, welding). Notably, in contrast to the proposition by [18], capturing images of objects is considered a physical task in our context due to its location-bound nature. However, using a camera or taking photos can also be classified as virtual tasks, especially for robot localization. These tasks can be executed independently or concurrently with other virtual tasks [17]. For the aforementioned reasons, we exclude parallelism between physical tasks, and in this work, we address the parallelism between one physical and one or more virtual tasks.

A. Multi-Robot Tasks

The concept of MR Tasks has been briefly described as tasks that demand the involvement of multiple robots for their execution [8]. However, the original MRTA taxonomy does not encompass the requisite task interdependencies inherent in MR tasks. Cross-Schedule or Complex Dependencies are deemed necessary when considering MR tasks [11]. We further elaborate on this definition by distinguishing between two types of MR tasks based on location and timing requirements. Firstly, there are tasks necessitating multiple robots to be present at the same location concurrently, referred to as MRLC tasks, for their successful execution. An example of an MRLC task is one requiring the collective effort of multiple robots to push a heavy object, which makes it infeasible for a single robot to move alone. An illustration is given in Fig. 2, where the MRLC task (t_{10}) can only begin when all 3 robots arrive at the task's location. The nature of interrelatedness in MRLC tasks can vary, depending upon the task's abstraction level [11].

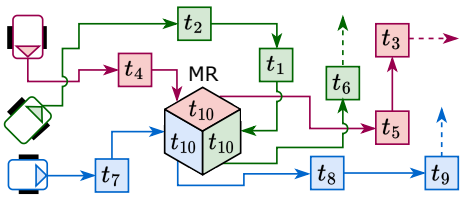

Fig. 2: An example of a plan including an MR Task (depicted as cube) requiring 3 robots. Different colors denote different agents.

Secondly, there are tasks mandating multiple robots to be present at the same location but not necessarily concurrently, and we refer to them as MRL tasks. An example of an MRL task is unlocking a locked door, where one robot is assigned to unlock the door, and another robot is tasked to open that door. Despite their complexity, such tasks can be modeled through a series of SR tasks, each with specific equipment requirements. Additionally, these tasks necessitate a precedence constraint between them, i.e., first, unlock the door, and only then can the door be opened. Generally, MRL tasks entail XD dependencies.

We can further categorize MR tasks based on equipment requirements into two categories. In MR Single-Equipment (MRSE) tasks, all robots involved require the same equipment to complete the task. For instance, when pushing a heavy box, as previously mentioned, all robots must possess the same capability to push the box. Conversely, in MR Multiple-Equipment (MRME) tasks, such as the door example mentioned earlier, robots are required to possess distinct capabilities. However, this work focuses solely on MRSE tasks, excluding MRME tasks from its scope. Thus, the primary focus of this work remains on MRLC and MRSE task categories, which we will simply refer to as MR for clarity. Generally, when discussing MR tasks, the assumption is that they are physical tasks, as the practical viability of virtual MR tasks is uncertain.

B. Mission Example

We present Fig. 3, which depicts a solution to a simple scenario involving 2 robots and 7 tasks (5 physical and 2 virtual). Only Task 2 requires multiple robots. Different task types are distinguished by color: physical tasks are orange, virtual tasks are blue, and MR tasks are pink. Transitions between physical task locations are shown in green at the bottom, with destination depots colored gray. In the figure, the start time of a task is marked with an s superscript and the end time with an e superscript. In this example, Task 6 is scheduled to run in parallel with Tasks 3 and 7, which is leveraged by the solver. Task 2 begins concurrently on both Robots 1 and 2. Upon completing Task 2, the robots proceed to the final depot δ.

IV. Problem Formulation

Consider an individual robot s belonging to a set of robots denoted as $\mathbb{S}$. The set $\mathbb{S} := \{s_1, s_2, \ldots, s_m\}$ comprises of m robots, each performing a set of tasks. These tasks are

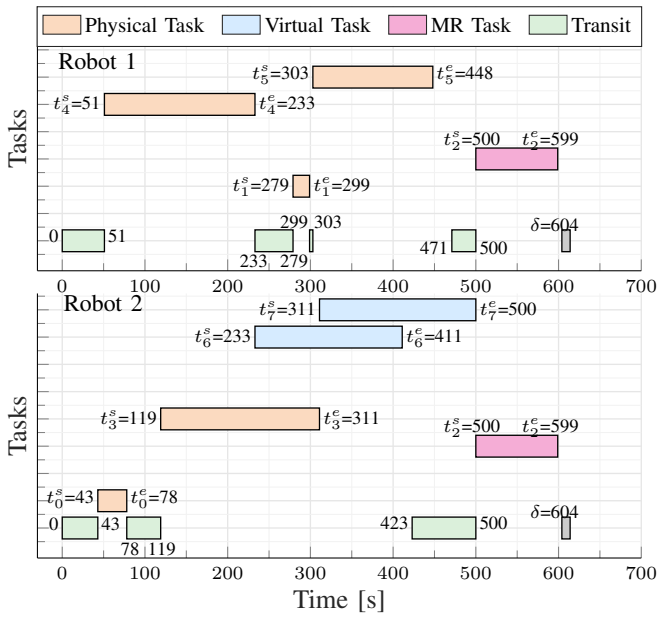

Fig. 3: Depiction of an optimal solution for a problem consisting of 2 robots and 7 tasks, out of which 4 are physical, 2 virtual, and 1 task is an MR task (task t_2). MR task is scheduled to be executed at the same time (500) on both robots 1 and 2.

part of the set $\mathbb{T} := \{t_1, t_2, \ldots, t_n\}$, which encompasses both physical and virtual tasks. Additionally, a collection of equipment types exists, expressed as $\mathbb{C} := \{c_1, c_2, \ldots, c_l\}$, consisting of l distinct equipment types. Each robot s is equipped with one or more of these equipment types c[1]. In this context, let σ represent a source depot from the set $\Sigma := \{\sigma_1, \sigma_2, \ldots, \sigma_q\}$, and δ a destination depot from the set $\Delta := \{\delta_1, \delta_2, \ldots, \delta_w\}$. The sets Σ and Δ contain q source depots and w destination depots. Each robot, denoted as s, commences its mission from a source depot in the set Σ and concludes its tour at a destination depot in the set Δ. Superset $\widetilde{\mathbb{T}}$ encompasses all the tasks as well as the source and destination depots, i.e., $\widetilde{\mathbb{T}} := \mathbb{T} \cup \Sigma \cup \Delta$. The number of elements in $\widetilde{\mathbb{T}}$ is denoted as $p = (q + n + w)$. A superset consisting of all the source depots and task elements is defined as $\mathbb{T}^{\Sigma} := \mathbb{T} \cup \Sigma$. Similarly, a superset comprising all the destination depots and tasks is defined as $\mathbb{T}^{\Delta} := \mathbb{T} \cup \Delta$.

Tasks are divided into virtual and physical tasks. We can define a set $\mathbb{H} = \{h_1, \ldots, h_n\}$, where $h_i = 1$ if task i is virtual, and 0 otherwise. To define which tasks can be performed in parallel, we define symmetrical matrix $\mathbf{R} = [r_{ijs}]_{p \times p \times m}$, where $i, j \in \widetilde{\mathbb{T}}$ and $s \in \mathbb{S}$. If tasks i and j can be done in parallel and $h_i + h_j \geq 1$, then $r_{ijs} = 1$. In this way, we also disallow the option of executing two physical tasks in parallel in any scenario. In addition, if either of i or j is in Σ or Δ, $r_{ijs} = 0$. We introduce set $\mathbb{K} := \{k_1, k_2, \ldots, k_n\}$, representing the number of robots required to complete each task in set $\mathbb{T}$. Each task i may require the number of robots in the range of $k_i \in \{1, \ldots, m\}$. In some cases, tasks i and j must be executed by the same robot s. To do this, we first define a (symmetric) matrix $\mathbf{U} = [u_{ij}]_{[n \times n]}$, whose elements are defined as $u_{ij} = 1$ if the same robot must do tasks i and j, and 0 otherwise. This entails that if $u_{ij} = 1 \implies \phi_c(i) = \phi_c(j)$.

It is important to recognize that both source and destination depots, in addition to tasks, are considered as nodes in the graph $\mathcal{G}(\widetilde{\mathbb{T}}, \mathbb{E})$, with $\mathbb{E}$ being the set of edges in the graph. Each edge e_{ij} connecting two nodes, i and j, in $\widetilde{\mathbb{T}}$ that is being traversed by robot s has a cost ω_{ijs}. When at least one of the nodes is virtual, regardless of the assigned robot, $\omega_{ijs} = 0$. We define cost matrix $\mathbf{\Omega} = [\omega_{ijs}]_{p \times p \times m}$ as symmetrical, i.e., $\omega_{ijs} = \omega_{jis}$.

Source depots function exclusively as root nodes in the graph, signifying that no edges can come from any other node to connect to a source depot. In contrast, destination depot nodes act solely as terminal nodes in the graph, implying that they do not have outgoing edges. Thus, upon reaching a destination depot, a robot's path concludes. Multiple robots may terminate their missions at the same destination depot, or a destination depot may remain unvisited. Notably, source and destination nodes are instantaneous, meaning they have no duration and cannot be virtual. On the other hand, each task i within $\mathbb{T}$ has a duration $\xi_{is} \in \mathbb{Z}_{\geq 0}$, representing the time required for robot s to complete task i. As mentioned, nodes $i \in \Sigma \cup \Delta$ have their duration ξ_{is} set to 0 since they are associated with source or destination depots.

To maintain precision in the definitions, when a set includes elements beyond those belonging to the set of tasks denoted as $\mathbb{T}$, all elements in those sets are called nodes. However, if the elements solely represent a subset of $\mathbb{T}$, they will be referred to as tasks. Precedence relations among tasks are described by the adjacency matrix $\mathbf{\Pi} = [\pi_{ij}]_{n \times n}$, where $i, j \in \mathbb{T}$, and $\pi_{ij} = 1$ if task i has to end before task j starts, if there is no such requirement $\pi_{ij} = 0$. Precedence relations are global and not robot-specific.

It is assumed that matrices $\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{R}, \overline{\mathbf{A}}_s$, and $\mathbf{U}$, as well as sets $\tilde{\mathbb{T}}, \mathbb{S}, \mathbb{C}, \mathbb{H}$, and $\mathbb{K}$ are problem dependent and are known *a priori*.

A. Integer Linear Programming Model

The model has three decision variables x_{ijs}, τ_i, z_{ijs}. The binary decision variable $x_{ijs} \in \{0, 1\}$ defines if robot s travels from node i immediately to node j. In that case, $x_{ijs} = 1$; otherwise, it is equal to 0. The decision variable $\tau_i \in \mathbb{Z}_{\geq 0}$ represents the starting time of a node i. The starting time of source depots is always 0, i.e., if $i \in \Sigma \implies \tau_i = 0$. The starting time of a destination depot is defined as the time at which the last robot arrives at that depot. Lastly, $z_{ijs} \in \{0, 1\}$ is a binary variable indicating if node i starts before node j on robot s. If this is true, i.e., $\tau_i < \tau_j$ then $z_{ijs} = 1$, and 0 otherwise. In case $\tau_i = \tau_j \wedge x_{ijs} = 1 \implies z_{ijs} = 1$, and it is 0 otherwise. Furthermore, every task $i \in \mathbb{T}$ requires certain equipment $\phi_c(i)$ for its successful completion, with $\phi_c : \mathbb{T} \mapsto \mathbb{C}$. Each robot $s \in \mathbb{S}$ has a set of available equipment $\mathbb{C}_s \subseteq \mathbb{C}$. An equipment matrix $\overline{\mathbf{A}}_s$

[1]To simplify the problem, we assumed that each task requires only one piece of equipment. Tasks requiring multiple equipment would not impact the optimization process, as composite equipment types can be defined to preserve the one-equipment-per-task model.

defines which tasks can be performed by robot $s \in \mathbb{S}$, and its elements are

$$\overline{a}_{ijs} = \begin{cases} 1, (i \in \Sigma, j \in \Delta) \vee (i \in \Sigma, j \in \mathbb{T} \wedge \phi_c(j) \in \mathbb{C}_s) \vee \\ \quad (i \in \mathbb{T} \wedge \phi_c(i) \in \mathbb{C}_s, j \in \Delta) \vee \\ \quad (\phi_c(i), \phi_c(j) \in \mathbb{C}_s \wedge i, j \in \mathbb{T}), \\ 0, (i, j \in \Sigma) \vee (i, j \in \Delta). \end{cases}$$

Equipment constraints do not affect source and destination depot nodes. We are now set to present the problem constraints.

$$x_{iis} = 0, \quad \forall i \in \widetilde{\mathbb{T}}, \forall s \in \mathbb{S}, \tag{1}$$

$$x_{ijs} \leq \overline{a}_{ijs}, \ \forall i \in \mathbb{T}^{\Sigma}, \forall j \in \mathbb{T}^{\Delta}, \forall s \in \mathbb{S}, \tag{2}$$

$$x_{ijs} \leq z_{ijs}, \ \forall i \in \mathbb{T}^{\Sigma}, \forall j \in \mathbb{T}^{\Delta}, \forall s \in \mathbb{S}, \tag{3}$$

$$\sum_{s \in \mathcal{S}} \sum_{i \in \mathbb{T}^{\Sigma}} x_{ijs} = k_j, \ \forall j \in \mathbb{T}, \tag{4}$$

$$\sum_{s \in \mathcal{S}} \sum_{j \in \mathbb{T}^{\Delta}} x_{ijs} = k_i, \ \forall i \in \mathbb{T}, \tag{5}$$

$$\sum_{i \in \mathbb{T}^{\Sigma}} x_{ijs} = \sum_{k \in \mathbb{T}^{\Delta}} x_{jks}, \ \forall j \in \mathbb{T}, \forall s \in \mathbb{S}, \tag{6}$$

$$\sum_{i \in \Sigma} \sum_{j \in \mathbb{T}^{\Delta}} x_{ijs} = 1, \ \forall s \in \mathbb{S}, \tag{7}$$

$$\sum_{i \in \mathbb{T}^{\Sigma}} \sum_{j \in \Delta} x_{ijs} = 1, \ \forall s \in \mathbb{S}, \tag{8}$$

$$\sum_{s \in \mathbb{S}} (z_{ijs} + z_{jis}) \leq k_i, \ \forall i, j \in \mathbb{T}, \tag{9}$$

$$\sum_{s \in \mathbb{S}} (z_{ijs} + z_{jis}) = k_i, \ \forall i, j \in \mathbb{T}, u_{ij} = 1, \tag{10}$$

$$z_{iks} + z_{kjs} - z_{ijs} \leq 1, \quad \forall i \in \mathbb{T}^{\Sigma}, \forall j \in \mathbb{T}^{\Delta}, \forall k \in \mathbb{T}, \forall s \in \mathbb{S}, \tag{11}$$

$$\tau_i + \xi_{is} + \omega_{ijs} \cdot z_{ijs} \leq \tau_j, \ \forall i, j \in \mathbb{T}, \forall s \in \mathbb{S}, t_i \prec t_j, \tag{12}$$

$$\tau_i + \mathcal{P}_{ijs} \leq \tau_j + \mathcal{M} \cdot (1 - z_{ijs}), \quad \forall i \in \mathbb{T}^{\Sigma}, \forall j \in \mathbb{T}^{\Delta}, s \in \mathbb{S}, \tag{13}$$

$$\mathcal{P}_{ijs} = z_{ijs} \cdot (\omega_{ijs} + \xi_{is}) \cdot (1 - r_{ijs}), \quad \forall i \in \mathbb{T}^{\Sigma}, \forall j \in \mathbb{T}^{\Delta}, s \in \mathbb{S}. \tag{14}$$

First, Eq. (1) removes self-loops on the nodes. We prevent robots from visiting tasks with incorrect equipment Eq. (2) that is defined in the equipment matrix ($\overline{\mathbf{A}}_s$). The relation between x_{ijs} and z_{ijs} is expressed by Eq. (3), meaning that if a robot s visits a node i right before a node j ($x_{ijs} = 1$), then $z_{ijs} = 1$. If this is not the case, i.e., if node i is not visited before node j, then $z_{ijs} = 0$ and also $x_{ijs} = 0$. The case when $x_{ijs} = 0$ and $z_{ijs} = 1$ indicates that task j is done after task i, but not immediately after, by robot s. The number of robots that need to start (Eq. (4)) and finish (Eq. (5)) each task is defined by set $\mathbb{K}$. Forcing the robot that started a task to finish it is expressed by Eq. (6). The start of every tour has to be at a source depot (Eq. (7)), while the end must always be at one of the destination depots (Eq. (8)). Some robots can go directly from a source to a destination depot without performing tasks, i.e., $x_{ijs} = 1, i \in \Sigma, j \in \Delta$. This means that the robot is not used in the final plan. We introduce Eq. (9) to prevent possible cycles in variable z, i.e., if task i is performed before task j, then it can never be that task j is also performed before task i thus removing possible cycles. In some cases, tasks may require to be executed by the same robot, e.g., they may need to use the results produced by other tasks, like sending the previously collected data. Eq. (10) constrains allocating tasks i and j to one single robot if $u_{ij} = 1$. Note that Eq. (9) does not force two tasks to be executed by the same robot. To maintain the transitive property of variable z_{ijs}, Eq. (11) is introduced. It ensures that if a node i is visited before a node k, and node k is visited before node j, then node i must be visited before a node j. In the model, we include also precedence constraints between tasks Eq. (12), i.e., a task i must be completed before task j, $t_i \prec t_j$. Specifically, for every robot s such that $z_{ijs} = 1$, the earliest scheduled time for a task j (τ_j) is the sum of (i) the starting time of task i (τ_i), (ii) the duration of task i performed by robot s (ξ_{is}), and (iii) the cost of moving from node i to node j (ω_{ijs}). If $z_{ijs} = 0$, the travel distance ω_{ijs} is eliminated from the equation, allowing task j to begin immediately after task i is completed, which is equal to $\tau_i + \xi_{is}$. In contrast, when $z_{ijs} = 1$, it indicates that robot s performs both tasks i and j. The disjunctive constraint defined by Eq. (13), in conjunction with Eq. (9), ensures that no two tasks can overlap on the same robot unless it is allowed by task parallelism, expressed by matrix $\mathbf{R}$. In Eq. (13), $\mathcal{M}$ is a big integer number [21]. In case tasks i and j are not performed by the same robot, i.e., $z_{ijs} = 0$, or if either task i, task j, or both are virtual, and parallel execution is allowed (where $r_{ijs} = 1$), $\mathcal{P}_{ijs}$ becomes 0. With this definition of Eq. (14), a wide range of task relations can be covered. To solve the described problem, it is necessary to allocate all tasks in $\mathbb{T}$ to a set of available robots $\mathbb{S}$, avoiding task repetition while respecting equipment and precedence requirements and minimizing the mission time. The resulting optimization problem is

$$\begin{aligned} &\underset{x,z,\tau}{\text{minimize}} && \max_{i \in \Delta}(\tau_i) \\ &\text{subject to} && \text{Constraints (1)–(14),} \end{aligned}$$

where i is one of the depot nodes. We opted to minimize the mission's makespan (minMax). However, various objective functions can be used [9].

B. CP Model

The formulation of the model is adapted to the syntax of IBM CP Optimizer [14]. The handling of MR tasks involves partitioning them into specific individual SR tasks. These tasks are synchronized to start simultaneously from the same location. The original task set $\mathbb{T} := \{t_1, t_2, \ldots, t_n\}$ consists of n tasks, where the number of robots required per task is not shown. In the definition of the CP model, we will extend task set $\mathbb{T}$ into $\overline{\mathbb{T}} = \{t_1^{(1)}, \ldots, t_1^{(k_1)}, \ldots, t_n^{(1)}, \ldots, t_n^{(k_n)}\}$. It consists of $v = |\overline{\mathbb{T}}|$ elements, where each k_i denotes the number of robots required to complete task t_i. Decision Interval Variables (DIVs) related to source depots are fixed to allow the mission to start at time 0.

Now we can define allocation matrix $\mathbf{A}_{[v\times m]}$ with the rows representing tasks and columns representing robots. Each element a_{ij} is a DIV for allocating task i to robot j. Another matrix that we need to define is the allocation matrix $\mathbf{AD}_{[m\times w]}$, where each element ad_{ij} is a DIV for the allocation of the robot i to the destination depot j. A robot can be allocated to only one destination depot; hence, all ad_{ij} are set to **optional**. Optional means that it is up to the solver to decide if the variable should be **present** or **absent**. Transition distances between physical tasks, source, and destination depot are expressed with the matrix $\mathbf{M}$, which is equivalent to Ω defined in Sect. IV, where ω_{ijs} is the cost of transitioning from node i to node j with robot s. The transition matrix $\mathbf{M}$ must satisfy the triangular inequality. Virtual tasks are not part of the transition matrix since no transition is necessary for their execution. The indices of the matrix $\mathbf{M}$ are numbered according to the sequence $\{1,\dots,q+|\mathbb{PT}|+w\}$, where the first q indices are associated with the source depots, $|\mathbb{PT}|$ is the cardinality of the set of physical tasks. The last w indices are associated with the destination depots. Based on the transition matrix $\mathbf{M}$, we can define a set of state functions $f_i \in \mathbb{F}$, where each element f_i represents a state function of robot i. A state function consists of a set of non-overlapping intervals within which the function maintains a defined non-negative integer state. Between these intervals, the state of the function is undefined, typically due to transitions occurring between different states. For a more detailed description of modeling using constraint programming and ILOG CP Optimizer, please refer to [3], [12], [13].

Example: Let us assume $\overline{\mathbb{T}} = \{t_1^{(1)}, t_1^{(2)}, t_2^{(1)}, t_3^{(1)}, t_3^{(2)}\}$, $\mathbb{K} = \{2,1,2\}$, and $\mathbb{S} = \{1,2,3,4\}$. If, for the moment, we ignore task/robot equipment requirements, then Fig. 4 represents a feasible allocation of tasks $t_1, t_2,$ and t_3, out of which t_1 and t_3 require 2 robots for their execution, to robots $s_1, s_2, s_3,$ and s_4. The allocation is visually represented with green squares, showing that the DIVs $a_{1,2}$, $a_{2,1}$, $a_{3,3}$, $a_{4,2}$, and $a_{5,4}$ are set to **present**. This means that the MR task t_1 is allocated to robots s_2 and s_1 ($t_1^{(1)}$ to s_2 and $t_1^{(2)}$ to s_1), SR task $t_2^{(1)}$ to s_3, and MR task t_3 to robots s_2 and s_5 ($t_3^{(1)}$ to s_2 and $t_3^{(2)}$ to s_5). The allocation in matrix $\mathbf{A}$ is constrained by Eq. (15), which in conjunction with Eq. (17) prevents double allocation (e.g., $t_1^{(2)}$ to robot s_2). In the case of robot

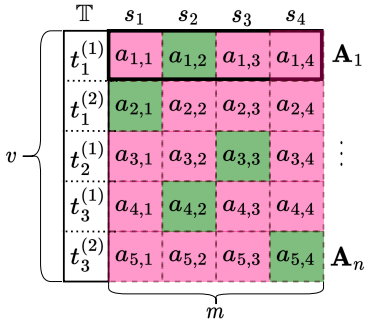

Fig. 4: The graphical representation of matrix $\mathbf{A}$. The green color marks **present** DIVs, and the pink **absent** DIVs.

s_j not having the necessary equipment to perform task $t_i^{(l)}$, the decision interval of that element is set to **absent**. For the sake of notation, let us introduce the following subsets of indices, $\mathbb{P}^s := \{1,\dots,q\}$, $\mathbb{P}^{pt} := \{q+1,\dots,q+|\mathbb{PT}|\}$, and finally $\mathbb{P}^d := \{q+|\mathbb{PT}|+1,\dots,q+|\mathbb{PT}|+w\}$.

$$\textit{Alternative}(\overline{\mathbb{T}}_i, \mathbf{A}_i),\ \forall i \in \{1,\dots,v\}, \tag{15}$$

$$\textit{Alternative}(\Delta_i, \mathbf{AD}_i),\ \forall i \in \{1,\dots,w\}, \tag{16}$$

$$\textit{NoOverlap}(a_{ik}, a_{jk}), \quad \forall i,j \in \{1,\dots,v\}, \forall k \in \{1,\dots,m\}, r_{ij} \neq 1, \tag{17}$$

$$\textit{EndBeforeStart}(\overline{\mathbb{T}}_i, ad_{jk}), \quad \forall i \in \{1,\dots,v\}, \forall j \in \{1,\dots,m\}, \forall k \in \{1,\dots,w\}, \tag{18}$$

$$\textit{EndBeforeStart}(\overline{\mathbb{T}}_i, \overline{\mathbb{T}}_j),\ \forall i,j \in \{1,\dots,v\}, \pi_{ij} = 1. \tag{19}$$

$$\textit{AlwaysEqual}(f_i, \sigma_i, \mathbb{P}_i^s),\ \forall i \in \{1,\dots,m\}, \tag{20}$$

$$\textit{AlwaysEqual}(f_i, ad_{ij}, \mathbb{P}_i^d), \quad \forall i \in \{1,\dots,m\}, \forall j \in \{1,\dots,w\}, \tag{21}$$

$$\textit{AlwaysEqual}(f_i, a_{ji}, \mathbb{P}_j^{pt}), \quad \forall i \in \{\forall s \in \mathbb{S} \mid \phi_c(j) \in \mathbb{C}_s\}, \forall j \in \mathbb{P}^{pt}, \tag{22}$$

$$\textit{PresenceOf}(a_{ik}) = \textit{PresenceOf}(a_{jk}), \quad \forall i,j \in \{1,\dots,v\}, \forall k \in \{1,\dots,m\}, u_{ij} = 1, \tag{23}$$

$$\textit{StartOf}(t_i^{(j)}) = \textit{StartOf}(t_i^{(l)}), \quad \forall i \in \{1,\dots,v\}, \forall j,l \in \{1,\dots,k_i\}. \tag{24}$$

The constraint denoted by (15) ensures that each task in the set $\overline{\mathbb{T}}$ is exclusively assigned to only one robot from the available set $\mathbf{A}_i$, which represents i-th row of matrix $\mathbf{A}$. In other words, for every task in $\overline{\mathbb{T}}$, a single decision variable a_{ij} is chosen from the robot set $\mathbf{A}_i$, indicating the specific assignment of task $\overline{\mathbb{T}}_i$ to a robot s_j. For instance, when the solver sets the decision variable $a_{1,3}$ to the value **present**, it indicates the allocation of task 1 to robot 3. Constraint (16) states that every robot ends its tour at one of the destination depots. For example, suppose the solver sets the decision interval variable $ad_{2,2}$ to **present**. In that case, it means that robot 2 is ending its tour in destination depot 2. Constraint (17) states that neither SR nor MR tasks can run in parallel on individual robots except when it is allowed by matrix $\mathbf{R}$, i.e., task i and task j can run in parallel only if $r_{ijs} = 1$. All tasks assigned to a robot must be done by that robot before the destination depot is reached (Eq. (18)). Precedence constraints are defined on the task set $\mathbb{T}$ instead of $\overline{\mathbb{T}}$ because it is assumed that MR tasks have the same precedence rules for every instance (k_i) in the expanded $\overline{\mathbb{T}}$. These constraints are imposed with Eq. (19), where $\mathbb{T}_i$ must end before $\mathbb{T}_j$ starts, and in conjunction with Eq. (24) all MR tasks will respect the imposed precedence constraint. To enforce the robots' position at the source depot at the start of their mission, we introduce Eq. (20). To ensure that the robots are positioned at one of the destination depots upon completing their missions, we introduce Eq. (21). Constraint (22) uses a state function to model the positions of robots with respect to physical SR or MR tasks, where the set $\{\forall s \in \mathbb{S} | \phi_c(j) \in \mathbb{C}_s\}$ is the set of robots that can execute the task j that requires the equipment $\phi_c(j)$, as they

have the correct equipment $\mathbb{C}_s$. In case the same robot is required to perform both tasks i and j, constraint (23) can be used. Constraint (24) forces all robots assigned to the same MR task to start executing that task simultaneously. The optimization objective is the same as in the ILP model, and it is the minimization of the mission's makespan.

V. Evaluation

The assessment involves evaluating a collection of test instances using two distinct solvers: the CPLEX solver and the CP Optimizer (CPO). The experimental infrastructure comprises an i9-9980XE @4.1GHz CPU with 18 cores and 128GB of DDR4 RAM. The solvers are configured with a timeout limit of 1 hour (3600 seconds). We aim to analyze and discern the differences among the solutions these two solvers generate. The benchmark consists of 30 problem instances with different levels of complexity regarding the number of robots, tasks, and constraints. We limit the number of different equipment required by tasks to 4. The number of precedence constraints depends on the number of virtual tasks in the problem instance. Specifically, all virtual tasks are set to succeed some physical tasks to avoid having virtual tasks executed at the beginning of the mission. In practice, virtual tasks commonly rely on the completion of physical tasks, e.g., performing data analysis virtually requires data acquisition during some physical tasks.

The task duration is set to range from 10 to 200 seconds. The test instances are created randomly, where tasks and robots are placed in a confined space of 200×200 meters. The likelihood of a task being a virtual task is set to 15%. The probability of tasks running in parallel is set to 50%, where we allow mixed parallelism between one physical and possibly multiple virtual tasks. A coin flip is used to assign each task as either SR or MR, ensuring that every test instance contains at least one MR task and no more than $(n-1)$ MR tasks. The number of robots a task may require is between 1 and 3. Virtual tasks cannot be MR tasks. For simplicity, matrix $\mathbf{R}$ is the same for all robots.

A. Results

The benchmark results are presented in Table I. Presented results include the makespan of the best solution found, the gap between the best solution and the best lower bound found, the time taken to find the first feasible solution (t-first), the best solution (t-best), and the time taken to prove optimality (t-total). The gap is defined as $gap = ((bs - bb)/bs) \cdot 100\%$, where bs is the best solution found, and bb is the best bound on the optimal solution [15]. For the first 4 instances, which include only 2 robots, both solvers have found optimal solutions in a comparable amount of time. CPO starts to take the lead in the following instances, as it can find solutions faster. In the first 10 instances, both solvers could find optimal solutions; however, CPLEX took much more time. CP optimizer dominated in all three time metrics. After the initial 10 instances, CPLEX matched the solution quality of the CP method only in instance 14. However, CPLEX required substantially more computational time to achieve this result. In every other instance, CPO was able to outperform CPLEX in terms of solution quality. Moreover, CPLEX could not find any feasible solution for instances 17 and 20–30 within the given time limit of 1 hour.

Each task set is repeated with varying parameters to demonstrate that even with the same number of tasks and robots, the problem's complexity can increase for the solvers due to differing constraints. For instance, this is evident in instances 7 and 8. Instance 7 is solved relatively easily compared to instance 8, which took almost 100x longer for the CPLEX solver. For the CPO solver, this difference is smaller, with instance 8 taking approximately 9x longer to solve. This difference is primarily due to the number of MR tasks: instance 7 has 2 MR tasks, whereas instance 8 has 4.

When it comes to proving optimality, neither of the solvers performed well on the harder instances, with CPO having a gap between 47-82% and CPLEX either proved a solution is optimal or had a gap of 79-96% However, this is somewhat expected as the solvers were given a fairly limited amount of time for the search (3600 seconds), which proved not to be enough to perform an exhaustive search. We limited the size of our test instances to a maximum of 6 robots and 18 tasks in the presented results. For this particular problem, given the presented ILP and CP models in this paper, CPO is a clear winner, as it is able to find equally good or better solutions than CPLEX for every instance. The time column t-best shows the time a solver takes to reach the best makespan. The rest of the time was spent by the solvers trying to prove the optimality of the found solution. If the gap is 0%, it means the found solution is proven to be optimal. Additionally, the CPO found a feasible solution for each instance in under 1 second, demonstrating its ability to find feasible solutions rapidly. This capability could be highly beneficial in scenarios requiring quick mission replanning due to unexpected events during execution.

VI. Conclusion

Recent technological advancements have created opportunities for employing Multi-Robot Systems across various scenarios, particularly those involving Multi-Task (MT) robots and Multi-Robot (MR) tasks. This study aimed to address a gap in the existing literature by formally defining the challenges associated with Multi-Task Robots, Multi-Robot Tasks, and Time-extended Assignment (MT-MR-TA), which have not been adequately explored in current surveys of Multi-Robot Task Allocation (MRTA) problems.

We approached this problem using Integer Linear Programming (ILP) and Constraint Programming (CP). Both models were implemented in commercially available solvers, CPLEX and CPO. Our evaluation indicates that the CP Optimizer is generally more effective for this type of problem configuration, as it outperformed CPLEX in most of the benchmark test instances. Moreover, CP formulation proves to be more appropriate for MT-MR-TA problems, which are highly constrained, due to its greater flexibility in managing combinatorial constraints, including task precedence, synchronization, and logical constraints.

TABLE I: CPLEX and CPO benchmark results. Better solutions are in bold. t-**total** is the solver's time to find a guaranteed optimal solution (the time limit is set to 3600). t-**first** is the solver's time to find the first feasible solution. t-**best** is the solver's time to find the best solution. Instances and code are available online: `https://github.com/mdh-planner/MT-MR-TA-Problem-Configuration`

	CPLEX					**CPO**				
Inst.	**Makespan**	**Gap**	t-**total**	t-**first**	t-**best**	**Makespan**	**Gap**	t-**total**	t-**first**	t-**best**
(1) 2R, 6T	332	0%	**0.44**	**0.07**	**0.36**	332	0%	2.9	0.09	0.48
(2) 2R, 6T	360	0%	**1.23**	**0.07**	0.48	360	0%	6.9	0.09	**0.1**
(3) 2R, 7T	282	0%	1.56	0.07	1.21	282	0%	**1.3**	**0.09**	**0.11**
(4) 2R, 7T	474	0%	**0.72**	0.06	0.72	474	0%	9.7	**0.1**	**0.11**
(5) 2R, 8T	301	0%	12	0.08	11.5	301	0%	**1.1**	**0.1**	**0.24**
(6) 2R, 8T	448	0%	274	0.12	11.3	448	0%	**72**	**0.13**	**0.44**
(7) 3R, 8T	242	0%	33	0.15	7.8	242	0%	**12.6**	**0.12**	**0.14**
(8) 3R, 8T	356	0%	2874	7	219	356	0%	**111**	**0.19**	**0.21**
(9) 3R, 10T	377	0%	817	0.28	2.4	377	0%	**16.7**	**0.12**	**0.14**
(10) 3R, 10T	305	0%	88.6	0.28	6.1	305	0%	**16.9**	**0.12**	**0.19**
(11) 3R, 12T	394	91%	3600	0.6	1768	**368**	**67%**	3600	**0.17**	**0.75**
(12) 3R, 12T	469	94%	3600	12.3	2892	**381**	**59%**	3600	**0.22**	**2.8**
(13) 4R, 10T	366	85%	3600	11.7	1088	**334**	**60%**	3600	**0.24**	**0.36**
(14) 4R, 10T	271	79%	3600	25	3353	271	**53%**	3600	**0.16**	**14.5**
(15) 4R, 12T	671	96%	3600	19.5	917	**418**	**82%**	3600	**0.4**	**9.9**
(16) 4R, 12T	462	92%	3600	3223	3549	**320**	**65%**	3600	**0.27**	**4**
(17) 4R, 14T	/	/	3600	/	/	**339**	**67%**	3600	**0.37**	**36.6**
(18) 4R, 14T	787	96%	3600	17.5	1445	**605**	**79%**	3600	**0.65**	**9.9**
(19) 5R, 12T	432	95%	3600	1281	3138	**262**	**47%**	3600	**0.43**	**1651**
(20) 5R, 12T	/	/	3600	/	/	**302**	**56%**	3600	**0.35**	**64.3**
(21) 5R, 14T	/	/	3600	/	/	**266**	**60%**	3600	**0.36**	**10.7**
(22) 5R, 14T	/	/	3600	/	/	**426**	**67%**	3600	**0.6**	**2.7**
(23) 5R, 16T	/	/	3600	/	/	**395**	**71%**	3600	**0.7**	**14.7**
(24) 5R, 16T	/	/	3600	/	/	**468**	**78%**	3600	**0.91**	**41.9**
(25) 6R, 14T	/	/	3600	/	/	**321**	**66%**	3600	**0.8**	**952**
(26) 6R, 14T	/	/	3600	/	/	**291**	**60%**	3600	**0.63**	**42.4**
(27) 6R, 16T	/	/	3600	/	/	**303**	**60%**	3600	**0.7**	**22.1**
(28) 6R, 16T	/	/	3600	/	/	**356**	**69%**	3600	**0.96**	**225**
(29) 6R, 18T	/	/	3600	/	/	**331**	**57%**	3600	**0.99**	**54.8**
(30) 6R, 18T	/	/	3600	/	/	**548**	**69%**	3600	**0.66**	**11.3**

REFERENCES

[1] E. A. Ameri, B. Cürüklü, B. Miloradović, and M. Ektröm. Planning and supervising autonomous underwater vehicles through the mission management tool. In *Global Oceans 2020: Singapore – U.S. Gulf Coast*, pages 1–7, 2020.

[2] H. Aziz, H. Chan, Á. Cseh, B. Li, F. Ramezani, and C. Wang. Multi-robot task allocation-complexity and approximation. In *Proceedings of the 20th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 133–141, 2021.

[3] R. Barták. Constraint programming: In pursuit of the holy grail. In *Proceedings of the Week of Doctoral Students (WDS99)*, volume 4, pages 555–564. MatFyzPress Prague, 1999.

[4] E. Bischoff, F. Meyer, J. Inga, and S. Hohmann. Multi-robot task allocation and scheduling considering cooperative tasks and precedence constraints. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3949–3956. IEEE, 2020.

[5] H. Chakraa, F. Guérin, E. Leclercq, and D. Lefebvre. Optimization techniques for multi-robot task allocation problems: Review on the state-of-the-art. *Robotics and Autonomous Systems*, 2023.

[6] M. Dadvar, S. Moazami, H. R. Myler, and H. Zargarzadeh. Exploration and coordination of complementary multirobot teams in a hunter-and-gatherer scenario. *Complexity*, 2021(1):9087250, 2021.

[7] A. Dorri, S. S. Kanhere, and R. Jurdak. Multi-agent systems: A survey. *IEEE ACCESS*, 6:28573–28593, 2018.

[8] 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*, 23(9):939–954, 2004.

[9] M. Gini. Multi-Robot Allocation of Tasks with Temporal and Ordering Constraints. In *Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, 2017.

[10] S. Hartmann and D. Briskorn. An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of OR*, 297(1):1–14, 2022.

[11] G. A. Korsah, A. Stentz, and M. B. Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12):1495–1512, 2013.

[12] P. Laborie and J. Rogerie. Reasoning with conditional time-intervals. In *21st International FLAIRS Conference*, pages 555–560, 2008.

[13] P. Laborie, J. Rogerie, P. Shaw, and P. Vilím. Reasoning with conditional time-intervals. part II: An algebraical model for resources. In *22nd International FLAIRS Conference*, 2009.

[14] P. Laborie, J. Rogerie, P. Shaw, and P. Vilím. IBM ILOG CP optimizer for scheduling: 20+ years of scheduling with constraints at IBM/ILOG. *Constraints*, 23:210–250, 2018.

[15] G. Laporte and P. Toth. A gap in scientific reporting. *4OR*, 20(1):169–171, 2022.

[16] B. Miloradović, B. Curuklu, M. Ekström, and A. V. Papadopoulos. Tamer: Task allocation in multi-robot systems through an entity-relationship model. In *International Conference on Principles and Practice of Multi-Agent Systems*. Springer, 2019.

[17] B. Miloradović, B. Çürüklü, M. Ekström, and A. V. Papadopoulos. Optimizing parallel task execution for multi-agent mission planning. *IEEE Access*, 11:24367–24381, 2023.

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

[19] E. Raboin, P. Švec, D. S. Nau, and S. K. Gupta. Model-predictive asset guarding by team of autonomous surface vehicles in environment with civilian boats. *Autonomous Robots*, 38:261–282, 2015.

[20] E. Raboin, P. Švec, D. Nau, and S. K. Gupta. Model-predictive target defense by team of unmanned surface vehicles operating in uncertain environments. In *2013 IEEE International Conference on Robotics and Automation*, pages 3517–3522, 2013.

[21] P. A. Rubin. Heuristic solution procedures for a mixed-integer programming discriminant model. *Managerial and Decision Economics*, 11(4):255–266, 1990.

[22] J. Snauwaert and M. Vanhoucke. A classification and new benchmark instances for the multi-skilled resource-constrained project scheduling problem. *European Journal of Operational Research*, 307:1–19, 2023.