IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 January 2023, accepted 5 March 2023, date of publication 9 March 2023, date of current version 15 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3254900

RESEARCH ARTICLE

Optimizing Parallel Task Execution for Multi-Agent Mission Planning

BRANKO MILORADOVIĆ, (Student Member, IEEE), BARAN ÇÜRÜKLÜ, MIKAEL EKSTRÖM, (Senior Member, IEEE), AND ALESSANDRO VITTORIO PAPADOPOULOS, (Senior Member, IEEE)
Division of Intelligent Future Technologies, Mälardalen University, 722 20 Västerås, Sweden

Corresponding author: Branko Miloradović (branko.miloradovic@mdu.se)

This work was supported in part by the Swedish Research Council (VR), through the Project Pervasive Self-Optimizing Computing Infrastructure (PSI); in part by the Knowledge Foundation (KKS), through the Project Dependable Platforms for Autonomous Systems and Control (DPAC) and Project Federated Choreography of an Integrated Embedded Systems Software Architecture (FIESTA); and in part by the European Commission, through the Project Aggregated Farming in the Cloud (AFarCloud).

ABSTRACT Multi-agent systems have received a tremendous amount of attention in many areas of research and industry, especially in robotics and computer science. With the increased number of agents in missions, the problem of allocation of tasks to agents arose, and it is one of the most fundamental classes of problems in robotics, formally known as the Multi-Robot Task Allocation (MRTA) problem. MRTA encapsulates numerous problem dimensions, and it aims at providing formulations and solutions to various problem configurations, i.e., complex multi-agent missions. One dimension of the MRTA problem has not caught much of the research attention. In particular, problem configurations including Multi-Task (MT) robots have been neglected. However, the increase in computational power, in robotic systems, has allowed the utilization of parallel task execution. This in turn had the benefit of allowing the creation of more complex robotic missions; however, it came at the cost of increased problem complexity. Our contribution to the aforementioned domain can be grouped into three categories. First, we model the problem using two different approaches, Integer Linear Programming and Constraint Programming. With these models, we aim at filling the gap in the literature related to the formal definition of MT robot problem configuration. Second, we introduce the distinction between physical and virtual tasks and their mutual relationship in terms of parallel task execution. This distinction allows the modeling of a wider range of missions while exploiting possible parallel task execution. Finally, we provide a comprehensive performance analysis of both models, by implementing and validating them in CPLEX and CP Optimizer on the set of problems. Each problem consists of the same set of test instances gradually increasing in complexity, while the percentage of virtual tasks in each problem is different. The analysis of the results includes exploration of the scalability of both models and solvers, the effect of virtual tasks on the solvers' performance, and overall solution quality.

INDEX TERMS Multi-agent mission planning, multi-robot task allocation, parallel task execution, integer linear programming, constraint programming.

I. INTRODUCTION

Multi-Agent[1] Systems (MASs) have been widely present in the robotics domain in the application areas of navigation, cooperation, and planning [1], [2]. The increase in the number of robots, and their capabilities, have led to the possibility to define more complex robotic missions, which is also desirable from the perspective of the user, or the problem owner. A mission usually consists of smaller segments, which are, in this context, represented as atomic tasks. The process of mapping tasks to robots is usually referred to as the Multi-Robot Task Allocation (MRTA) problem.

To our knowledge, the first formal definition in its current scope is proposed by Gerkey and Matarić [3]. This problem represents one of the most fundamental

The associate editor coordinating the review of this manuscript and approving it for publication was Okyay Kaynak.

[1]In this paper, we use the terms *robot* and *agent* interchangeably.

classes of problems in robotics and has been an active research area in different forms for many decades together with the evolution of the robotics domain, consequently leading to numerous research papers and MRTA taxonomy extensions [4], [5].

The robot count in a mission is not the only factor that defines a mission to be considered complex. The task account, and the number of constraints describing the mutual relations between them and the agents, play a pivotal role in determining the complexity of a robotic mission. Note that these factors are directly related to the process of producing a mission plan, which is performed by algorithms. There are other factors, such as the environmental setting, and changes in that environment during a mission.

The process of allocating tasks to agents with respect to agents' capabilities and task requirements is called Multi-Agent Mission Planning and is, at its core, equivalent to the MRTA problem. Although it is possible to let a human manually plan a mission for very small instances, solving the MRTA problem efficiently requires automated planning algorithms to be able to deal with a high number of tasks and constraints; however, even automated planning falls short in terms of scalability [6]. It is important to note, that in this work the focus is on the high level of abstraction in the planning domain. While task decomposition and path planning [7], [8], [9] are an important part of mission planning they are out of the scope of this work, as they are considered in the mid-level planning or even in low-level planning by being directly performed on agents.

As the robots are deployed in the environment where their allocated tasks are found, their goal is to perform their tasks as efficiently as possible. In order to achieve this objective, the mission planner must optimize both the routes that robots need to take and the times at which tasks will be performed, depending on their mutual interrelatedness. Hence, this problem can be described as a mixture of routing and scheduling problems. Even in their simple forms, both routing [10]) and scheduling [11] problems are NP-hard. Consequently, the combination of these two problems is also at least NP-hard.

From the computational perspective, robotics systems' capacities have become more powerful, over the years, thanks to the advances in modern parallel real-time computing technologies [12]), enabling an unprecedented level of parallelism, in terms of sensing, computation, motion, and manipulation tasks. As a result, it is plausible to assume that, more complex missions can be achieved by the robotics system, including tasks that require only computational capacity, and no physical actuation, e.g., data processing or data transmission [13]. The utilization of the additional computational tasks comes at the cost of increased complexity for the MRTA and, more in general, for the mission planning. However, possible benefits prove worth the extra complexity, as exploitation of the possible task parallelism may ultimately lead to a shorter mission duration and better usage of available hardware resources.

In this paper, we introduce a new distinction between physical and virtual tasks in the context of MRTA, and their relation in terms of parallel execution. This distinction captures the additional computational capabilities of robotics systems, and it allows for a more rich specification of the task set required to complete the mission. Moreover, following the proposed MRTA taxonomy, we propose a mathematical formalization of the mission planning problem denoted as MT-SR-TA. Where MT stands for Multi-Task robots, SR represents Single-Robot tasks, and TA assumes that we are dealing with Time-extended Assignment.

This paper has three main contributions:

1) formulation of two models of the MT-SR-TA problem configuration. The first one is in the form of an Integer Linear Programming (ILP) problem, and it is verified in the CPLEX optimization tool. The second model is a Constraint Programming (CP) model, implemented in the CP Optimizer. Both models are evaluated on a set of problem instances;
2) introduction of two task types – physical and virtual – based on their spatial constraints and discuss their temporal relations in the possible MT-SR-TA real-world scenario;
3) a comprehensive analysis of the proposed models in terms of solution quality and computational time.

A preliminary version of this work appeared in [14]. Additional contributions are: (i) formalization of the problem as a constraint programming model; (ii) comprehensive analysis of both ILP and CP models. The motivation behind the use of CP is to determine if it can outperform the ILP model implemented in CPLEX, as CPLEX was not able to efficiently solve many of the problem instances. It has been shown that for job shop scheduling problems, the CP model outperforms the ILP model by the order of magnitude [15]. As the problem described in this paper is a mixture of routing and scheduling problems, CP poses a viable option to outperform CPLEX.

The remainder of this paper is organized as follows. Sect. II gives an overview of the required background on MRTA problems. A discussion on different task types and how their nature affects the mission planning problem is given in Sect. III. The ILP formulation of the MT-SR-TA problem is presented in Sect. IV. The CP formulation of the same problem is given in Sect. V. Sect. VI describes a real-world case study of a multi-robot application and how it is solved by the proposed approach. Sect. VII discusses a more extensive evaluation of the presented models, implemented in CPLEX and CPO. Sect. VIII provides an overview of the related work. Finally, Sect. IX concludes the paper.

II. BACKGROUND

In the MRTA taxonomy for problem classification, Gerkey and Matarić [3] identified three main problem dimensions covering a total of 8 different problem configurations. This taxonomy includes:

- *Single-Task (ST)* vs. *Multi-Task (MT)* robots, distinguishing problems where robots can execute only a

single task at a time, from the ones where robots can execute multiple tasks at the same time;
- *Single-Robot (SR)* vs. *Multi-Robot (MR)* tasks, distinguishing tasks that require only a single or multiple robots to complete; and
- *Instantaneous Assignment (IA)* vs. *Time-extended Assignment (TA)*, distinguishing between robots having information only on the next task or being provided with a complete set of tasks, i.e., schedule.

As stated by the authors, there has been very little work done related to the problem configurations that include MT robots, especially the MT-SR-TA configuration. In an attempt to provide a reasonable approximation to the problem formulation, Gerkey and Matarić [3] state that the MT-SR-TA configuration can be seen as an ST-MR-TA with the set partitioning problem inverted, i.e., splitting a set of tasks into agent-specific coalitions. However, such a formulation does not consider the task locations and the time required for the robots to move between the tasks, thus the focus is only on the task allocation problem. Moreover, the interrelatedness of the tasks has not been addressed in [3], which makes the definition of MT or MR configurations incomplete.

The next notable contribution to the MRTA taxonomy was done by Korsah et al. [4] by proposing *iTax*, which is a framework aimed at addressing the task interrelatedness that was missing from the original MRTA taxonomy. Consequently, this work provides a survey on different problem configurations. Note that the MT configuration lacked a formal problem definition, and in general, this topic has not been investigated extensively by the research community. The authors state that some variants of the Vehicle Routing Problem (VRP), like pick-up and delivery, can be seen as an example of MT problem configuration. If we assume that, from the pick-up time until the delivery we have a continuously ongoing task, then every stop to pick up or deliver, other packages can be seen as a form of parallel task execution. The problem here is the level of abstraction of the tasks. If we assume that the task starts when the package is picked up and ends when that package is delivered, the VRP can be considered to have an MT-SR-TA configuration. However, it requires a high-level description, since the task is composed of sub-tasks. Thus, it would be more sensible to describe such a task as a set of three tasks, i.e., (1) pick-up of Package A, (2) transit from a pick-up to delivery location, and (3) delivery of Package A. In this scenario, performing other tasks between pick-up of Package A, and delivery of Package A, cannot be seen as a parallel task execution. Moreover, even if we assume these three tasks to be one monolithic task, still, tasks are not executed in parallel, but preemptively. For example, picking up some other Package B, before Package A has been delivered, can be seen as an interruption of Task A and not as parallel execution of Task A and B. For this reason, we discard the VRP as an example of an MT-SR-TA problem configuration. In addition, even the authors of this taxonomy, in the tables summing problems for each configuration, left MT-SR-TA as a blank field.

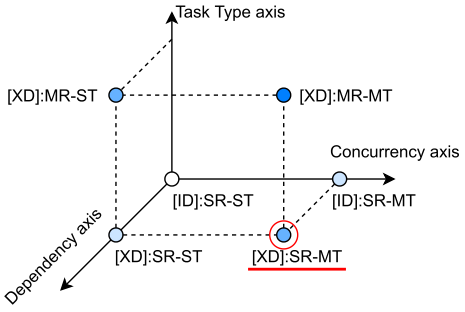

FIGURE 1. Marked in red is the problem configuration addressed in this paper, which lies on the *iTax* and MRTA axis.

From the *iTax*, we can conclude that if tasks have relations to other tasks within the same agent's schedule, the problem falls into the interrelatedness category of Intra-schedule Dependencies (ID). On the other hand, if there is a relation among tasks that are scheduled for execution on different agents, e.g., Precedence Constraints (PCs), there must exist a dependency between those schedules. This is called a Cross-schedule Dependency (XD). Other task dependencies are covered in the *iTax* as well, but they are out of the scope of this paper.

In the taxonomy extension done by Nunes et al. [5], two new dimensions have been added, a Synchronization Precedence (SP) and Time Windows (TW). The former refers to an ordering constraint, e.g., Task A needs to be completed before Task B, either by the same agent or a different agent; here the latter refers to a constraint that a certain task has to be performed in a predefined time slot. In this work, we focus on the SP dimension. In addition, Nunes et al. [5] provided a survey of the literature for other MRTA problem types. For the problem configuration that is of interest to us, MT-SR-TA, the conclusion was that this part of the MRTA problem remained unexplored. Even in the latest MRTA taxonomy extension by Miloradović et al. [16], the MT part has not been further explored, but it is rather kept as it was in the MRTA taxonomy.

As can be seen from the analysis of the most influential MRTA taxonomies, MT has been almost completely neglected as a research direction, both from a theoretical and practical perspective [3], [4], [5], [16]. This is still the case today.

The visualization of the relevant axes and arising problem configurations is presented in Fig. 1. This is a subset of *iTax* and original MRTA taxonomies and their dimensions. More precisely, we focus on the Task Type and Concurrency axis from the original MRTA taxonomy and Dependency axis of *iTax*. We omit the Assignment axis, as we assume that the problem we are addressing is always of Time-Extended Assignment type. In the figure, the problem configuration that is the focal point of this paper is marked in red. More specifically, this paper focuses on a special class of MT-SR-TA problems, i.e., the [XD]:MT-SR-TA-SP problem.

In the next section, a discussion on different types of tasks and their mutual relationship will be provided.

III. VIRTUAL AND PHYSICAL TASK TYPES

In the previous section, we have positioned the problem that this paper aims to address and framed it within the MRTA taxonomy. As it can be noted, the distinction between tasks mostly focuses on the required number of robots for successful task completion. We propose an additional task separation, i.e., we distinguish between two types of tasks that are denoted as *virtual* and *physical* tasks. Both virtual and physical tasks can also be SR or MR tasks.

Virtual tasks are defined as tasks that have no spatial constraints for their execution, i.e., they can be executed in any physical location, and with the level of parallelism allowed by the computing platform. For example, communicating and sending proprioceptive data, data analysis, etc. In general, these tasks can be performed as solo tasks, i.e., they do not overlap with any other tasks, during the execution of other tasks, or the transition from one task to the next.

On the other hand, physical tasks are defined as tasks that are bound to a certain location where they must be performed. Opposite to virtual tasks, physical tasks cannot be executed during the transition from one task to the next one. Physical tasks include physical manipulation of objects, environment sensing (e.g., scanning seabed, taking photos of objects of interest, etc.), or using tools (e.g., drilling, welding, etc.). Contrary to what has been proposed by Nunes et al. [5], we assume taking pictures of objects to be a physical task as well since it has to be done at a specific location. However, taking photos or using a camera, in general, can be a virtual task as well if it is used, e.g., for localization of the agent. These types of tasks are either performed as solo tasks or in parallel with some other task. Note that in this work, we do not address preemptive tasks.

The parallelism among tasks can be divided into three categories: (i) physical parallelism, among two or more physical tasks; (ii) virtual parallelism, among two or more virtual tasks; and (iii) mixed parallelism, among a mix of two or more tasks that can be either virtual or physical.

From a modeling standpoint, two or more physical tasks that can run in parallel must have the same location, therefore they can be modeled as a unique task associated with the given location, and with a duration computed either as the maximum of the two or more durations (in case they can all run in parallel) or by solving a local scheduling problem minimizing the makespan of the physical tasks. For example, a robot with two arms manipulating two different objects can be interpreted as parallel execution of two separate/independent tasks. Consequently, these tasks can be modeled as a single monolithic task, and the mapping between the multiple physical tasks to the single monolithic task is beyond the scope of this work.

The situation with the other two cases is, however, different. Virtual parallelism cannot always be achieved, e.g., due to functional dependencies among the tasks, the contention of the required resources, or simply because the level of parallelism provided by the computing platform is not enough to support the amount of concurrent virtual tasks. Similarly, also mixed parallelism may sometimes not be possible, for example, because of functional dependencies among the tasks, or because of physical limitations, e.g., scanning the seabed using sonar and using an acoustic modem for communication is not possible due to possible interference underwater.

The amount of physical or virtual tasks in a mission determines if the problem is closer to a routing or scheduling problem. For example, a mission with more physical tasks is closer to a routing problem, while a mission with more virtual tasks can be seen as closer to a scheduling problem. This ratio between the two task types may affect the performance of the algorithm in use (see Sect. VII for further elaboration).

IV. ILP MODEL

Before we present a mathematical formulation of the [XD]:MT-SR-TA-SP problem configuration, we will give a brief explanation of the problem.

The problem consists of allocating n tasks to m agents with respect to given constraints. Mainly, task requirements have to match the agent's capabilities in order for the task to be performed by that agent. In addition, there can be ordering constraints between tasks, meaning that some task has to be done before others can start. The objective of the optimization process is to minimize the overall makespan of the mission. The next section gives a detailed explanation of all variables and constraints used in creating the ILP model.

A. PRELIMINARIES AND NOTATION

Let $s \in \mathbb{S}$ be an agent, in a set $\mathbb{S} := \{s_1, s_2, \ldots, s_m\}$ of m agents that need to perform a set of n tasks, $\mathbb{T} := \{t_1, t_2, \ldots, t_n\}$; $\mathbb{T}$ includes both physical and virtual tasks. Also, let c be a type of equipment in a set $\mathbb{C} := \{c_1, c_2, \ldots, c_k\}$ of k types of equipment; each agent is endowed with one or more pieces of equipment, and every task requires one specific piece of equipment to be completed.

We denote with σ a source depot in a set $\Sigma := \{\sigma_1, \sigma_2, \ldots, \sigma_q\}$ of q source depots and, analogously, let δ be a destination depot in a set $\Delta := \{\delta_1, \delta_2, \ldots, \delta_w\}$, of w destination depots.

Each agent s starts from a source depot σ and finishes its tour at a destination depot δ. The superset containing all the tasks and the source and destination depots is defined as $\widetilde{\mathbb{T}} := \mathbb{T} \cup \Sigma \cup \Delta$. In addition, a superset containing all the source depots and tasks elements is defined as $\mathbb{T}^{\Sigma} := \mathbb{T} \cup \Sigma$. Analogously, a superset containing all the destination depots and tasks is defined as $\mathbb{T}^{\Delta} := \mathbb{T} \cup \Delta$. These supersets are defined for convenience, as they make the rest of the formulation more readable and compact. Both source and destination depots and tasks are nodes in the graph $\mathcal{G}(\widetilde{\mathbb{T}}, \mathbb{E})$, where $\mathbb{E}$ is the set of edges in the graph. Source depots (as well as destination depots) can be seen as "dummy" tasks, in the sense that they are still nodes in the graph, but special conditions apply to them. In the case of source depot, we can never have an edge in the graph leading to a source depot from any other node, thus source depots are always regarded as root nodes of the graph. In contrast, destination depot nodes can never have an edge leading out of the node. This means

that once the destination node is reached, it is the end of the path for the agent that reached it. Destination depot nodes differ from Task nodes as destination depots can be visited multiple times, e.g., all agents might end their missions in the same destination depot. Destination depot nodes may also remain unvisited. Both source and destination nodes are instantaneous, i.e., they do not have any duration compared to Task nodes. In addition, source and destination nodes assume the definition of physical tasks, i.e., they cannot be virtual. In order to maintain the definition correctness, when a set contains other elements besides the elements of the set $\mathbb{T}$, we will refer to all elements in those sets as nodes, while if the elements are only a subset of $\mathbb{T}$, they will be referred to as tasks.

Every edge $e(i, j)$ connecting two nodes $i, j \in \widetilde{\mathbb{T}}$, out of which at least one is virtual, regardless of the assigned agent, has a cost $\omega_{ijs} = 0$. The cost matrix $\Omega = [\omega_{ijs}]_{n\times n\times m}$ is symmetrical, i.e., $\omega_{ijs} = \omega_{jis}$.

The model has three decision variables x_{ijs}, τ_i, z_{ijs}. The decision variable $x_{ijs} \in \{0, 1\}$ defines if the agent s travels from node i to node j, and is defined as

$$x_{ijs} = \begin{cases} 1, & \text{if } s \in \mathbb{S} \text{ visits node } i \in \mathbb{T}^{\Sigma} \text{ immediately before} \\ & \text{node } j \in \mathbb{T}^{\Delta}, \\ 0, & \text{otherwise.} \end{cases}$$

The decision variable $\tau_i \in \mathbb{Z}_{\geq 0}$ defines the starting time of a node i. The decision variable $z_{ijs} \in \{0, 1\}$ is a binary variable that is indicating if a task i is started before a task j by an agent s. In other words, if $\tau_i \leq \tau_j$, and both tasks i and j are allocated to the agent s, $z_{ijs} = 1$. In any other case $z_{ijs} = 0$.

Every task $i \in \mathbb{T}$ has a duration $\xi_{is} \in \mathbb{Z}_{\geq 0}$, representing the amount of time agent s needs in order to complete task i; if node $i \in \Sigma \cup \Delta$, its duration ξ_{is} will be equal to 0, since it is associated with a source or destination depot, and no task needs to be performed.

Precedence relations among tasks are described by the adjacency matrix $\Pi = [\pi_{ij}]_{n\times n}$, where $i, j \in \mathbb{T}$, and

$$\pi_{ij} = \begin{cases} 1, & \text{if task } i \text{ is finished before task } j \text{ started, denoted} \\ & \text{as } i \prec j, \\ 0, & \text{otherwise.} \end{cases}$$

In addition, 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 agent $s \in \mathbb{S}$ has a set of available equipment $\mathbb{C}_s \subseteq \mathbb{C}$. An equipment matrix defines which tasks can be performed by agent s, and it is defined as

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

Equipment constraints do not affect the source and destination depot nodes, i.e., an agent s cannot have "wrong" equipment for depot nodes. In other words, an agent cannot violate equipment constraints in source and destination depots.

Lastly, matrix $\mathbf{R} = [r_{ij}]_{n\times n}$ can be defined, where $i, j \in \mathbb{T}$, which specifies which tasks can be performed in parallel

$$r_{ij} = \begin{cases} 1, & \text{if tasks } i \text{ and } j \text{ can be done in parallel,} \\ 0, & \text{otherwise.} \end{cases}$$

The matrix $\mathbf{R}$ is symmetrical.

This problem can be seen as a combination of a routing and scheduling problem. In the next section, we will introduce the routing part of the problem.

B. ROUTING PART OF THE FORMULATION

In the following, we present the constraints related to the routing part of the problem, with a brief explanation of what is their interpretation in natural language.

It is forbidden for an agent s to visit nodes with incorrect equipment, defined in the equipment matrix ($\mathbf{A_s}$):

$$x_{ijs} \leq \overline{a}_{ijs}, \qquad \forall i \in \mathbb{T}^{\Sigma}, \forall j \in \mathbb{T}^{\Delta}, \forall s \in \mathbb{S}. \tag{1}$$

Exactly one agent can start a task (Eq. (2)), and it can do it exactly once:

$$\sum_{s\in\mathbb{S}} \sum_{i\in\mathbb{T}^{\Sigma}} x_{ijs} = 1, \qquad \forall j \in \mathbb{T}, \tag{2}$$

Moreover, the agent that starts task j must also finish it:

$$\sum_{i\in\mathbb{T}^{\Sigma}} x_{ijs} = \sum_{k\in\mathbb{T}^{\Delta}} x_{jks}, \quad \forall j \in \mathbb{T}, \forall s \in \mathbb{S}. \tag{3}$$

The start of every tour has to be at a source depot (Eq. (4)), while the final destination must always be at one of the destination depots (Eq. (5)):

$$\sum_{i\in\Sigma} \sum_{j\in\mathbb{T}^{\Delta}} x_{ijs} = 1, \quad \forall s \in \mathbb{S}, \tag{4}$$

$$\sum_{i\in\mathbb{T}^{\Sigma}} \sum_{j\in\Delta} x_{ijs} = 1, \quad \forall s \in \mathbb{S}. \tag{5}$$

Note that some agents can go directly from a source depot to a destination depot without performing any of the tasks, i.e., $x_{ijs} = 1, i \in \Sigma, j \in \Delta$. This means that the agent is not used in the final plan.

Finally, self-loops on the nodes are forbidden:

$$x_{iis} = 0, \quad \forall i \in \widetilde{\mathbb{T}}, \forall s \in \mathbb{S}, \tag{6}$$

With these constraints, the routing part of the problem has been defined, and it is based on the definition of CTSP and ECTSP problems [17], [18]. The next section presents the scheduling part of the problem. It is important to note that the routing part of the problem, on its own, does not remove possible sub-tours from appearing in the solution. However, in the combination with the scheduling constraints, sub-tours are eliminated from the final solution.

C. SCHEDULING PART OF THE PROBLEM

In the following section, we present the constraints related to the scheduling part of the problem, with a brief explanation of what is their interpretation in natural language. The relation between x_{ijs} and z_{ijs} is expressed as:

$$x_{ijs} \leq z_{ijs}, \quad \forall i \in \mathbb{T}^{\Sigma}, \forall j \in \mathbb{T}^{\Delta}, \forall s \in \mathbb{S}, \tag{7}$$

meaning that if an agent 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 agent s. We introduce Eq. (8) 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.

$$\sum_{s\in\mathbb{S}}(z_{ijs} + z_{jis}) \leq 1, \qquad \forall i, j \in \mathbb{T}. \tag{8}$$

Eq. (9) maintains the transitive property of variable z_{ijs}. 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.

$$z_{iks} + z_{kjs} - z_{ijs} \leq 1, \ \forall i \in \mathbb{T}^{\Sigma}, \forall j \in \mathbb{T}^{\Delta}, \forall k \in \mathbb{T}, \forall s \in \mathbb{S}. \tag{9}$$

In the model, we include also precedence constraints between tasks, i.e., a task i must be completed before task j, $i \prec j$. Specifically, for every agent s such that $z_{ijs} = 1$, the earliest schedule time for a task j is the sum of: (i) the starting time of the node i (τ_i), (ii) the duration of the node i (ξ_{is}), and (iii) the cost of moving from node i to task j (ω_{ijs}), i.e.:

$$\tau_i + \xi_{is} + \sum_{s\in\mathbb{S}}(\omega_{ijs} \cdot z_{ijs}) \leq \tau_j, \ \forall i \in \mathbb{T}^{\Sigma}, \forall j \in \tilde{\mathbb{T}}, \forall s \in \mathbb{S}, i \prec j. \tag{10}$$

The sum of z_{ijs} over all the agents is equal to 0 in the case when task i and task j are not performed by the same agent, thus removing the travel distance ω_{ijs} from the equation. In contrast, when the sum of z_{ijs} over all the agents is equal to 1 it indicates that agent s performs both tasks i and j.

The disjunctive constraint defined by Eq. (11), in conjunction with Eq. (8), ensures that no two tasks can overlap on the same agent unless it is allowed by their parallelism, expressed by the matrix $\mathbf{R}$:

$$\tau_j + \mathcal{M} \cdot (1 - \sum_{s\in\mathbb{S}} z_{ijs}) \geq \tau_i + \mathcal{P}_{ij}, \quad \forall i \in \mathbb{T}^{\Sigma}, \forall j \in \tilde{\mathbb{T}}, \tag{11}$$

where $\mathcal{M}$ is a big integer number, and $\mathcal{P}_{ij}$ is a non-negative real number that expresses the time difference between the starting time of task i (τ_i) and the starting time of task j (τ_j).

$$\mathcal{P}_{ij} = \sum_{s\in\mathbb{S}} z_{ijs} \cdot (\omega_{ijs} + \xi_{is}) \cdot (1 - r_{ij}), \quad \forall i \in \mathbb{T}^{\Sigma}, j \in \mathbb{T}^{\Delta}, s \in \mathbb{S}. \tag{12}$$

In case tasks i and j are not performed by the same agent, i.e., $\sum_{s\in\mathbb{S}} z_{ijs} = 0$, $\mathcal{P}_{ij}$ is also equal to 0. $\mathcal{P}_{ij}$ also has the value of 0 in the case where either task i, or j, or both are virtual and parallel execution is allowed, i.e., $r_{ij} = 1$.

With this definition of $\mathcal{P}_{ij}$ a wide range of task relations can be covered, both physical and virtual. This will be demonstrated in Sect. VI on a case study for multi-robot agent planning, and in Sect. VII, on a range of different problem instances.

In some cases, tasks may require to be executed by the same agent, e.g., they may need to use the results produced by other tasks, like sending the data that was previously collected. If this happens, a new constraint needs to be added. Therefore, let $\mathbb{U}_{[n\times n]}$ be a (symmetric) matrix, whose elements are defined as $u_{ij} = 1$ if tasks i,j must be executed by the same agent, and $u_{ij} = 0$ otherwise. Thus, the constraint can be expressed as:

$$\sum_{s\in\mathbb{S}}(z_{ijs} + z_{jis}) = 1, \qquad \forall i, j \in \mathbb{T}, u_{ij} = 1, \tag{13}$$

constraining the allocation of both tasks i and j to one single agent. Note that Eq. (8) does not force two tasks to be executed by the same agent.

Constraints (7)–(13) define the scheduling part of the problem and in conjunction with constraints (1)–(6) formulates the ILP model of the XD:SR-MT-TA problem configuration.

D. OPTIMIZATION PROBLEM

To solve the previously described problem it is necessary to allocate all tasks in $\mathbb{T}$ to a set of available agents $\mathbb{S}$, avoiding task repetition, while respecting equipment and precedence requirements, and minimizing the mission time. The resulting optimization problem can be expressed as:

$$\begin{aligned} &\underset{x,z,\tau}{\text{minimize}} && \max_{i\in\Delta}(\tau_i) \\ &\text{subject to} && \text{Constraints (1)–(13).} \end{aligned}$$

Gini [19] defines a collection of possible objective functions to be used depending on the mission requirements. We chose to minimize the makespan of the mission.

Although ILP representation for these kinds of problems is very common, it is not the only possible approach. In the next section, we will present a constraint programming model in order to compare both models and solvers. The results of this comparison are given in Sect. VII.

V. CP MODEL

Now, we will present the problem formulation in a form of constraint programming. The formulation of the model is adapted to the solver (IBM CP Optimizer) syntax. For a more detailed description of modeling using Constraint Programming and ILOG CP Optimizer, please refer to [20] and [21]. However, for completeness, we will briefly introduce the main concepts used in this model.

Interval Variable *dvar* is a decision variable that has a domain of $dom(dvar) = \{\perp\} \cup \{[st, et) | st, et \in \mathbb{Z}_{\geq 0}, st \leq et\}$, where st defines the task's start time and et the task's end time. The difference of $(et - st) = td$ represents the task duration. An interval variable can be **optional**, i.e., it can be left up to the solver to decide if the variable is **absent** or **present**. If we define a fixed interval variable as $\underline{dvar}$, then, in the case the variable is absent, $\underline{dvar} = \perp$, and in the case the variable is present, $\underline{dvar} = [st, et)$. This is a very important property of interval variables that allows tasks not

to be performed if an alternative task is chosen. Most of the variables used in this model will be interval variables.

Alternative Constraint models an alternative amongst several optional variables. It is defined as $\{dvar, \{dvar_1, \ldots, dvar_n\}\}$; if an interval variable $dvar$ is present, then exactly one of intervals $\{dvar_1, \ldots, dvar_n\}$ is present. Moreover, $dvar$ starts and ends at the same time as the chosen variable in the interval. This constraint is used in our model to express the allocation of tasks to agents and the allocation of agents to destination depots.

Precedence Constraint models different ways of precedence. The one that is of interest to us is the precedence between the end time of one interval and the start time of another one (in CPO $endBeforeStart(dvar_1, dvar_2)$). We use this constraint in the model to define the relations between two tasks, where one has to end before the other one can start.

NoOverlap Constraint disallows any given pair of interval variables to overlap. Since our model deals with multitasks, overlapping is defined by the matrix $\mathbf{R}$, and it is related only to tasks assigned to the specific agent and not across all agents.

AlwaysEqual Constraint returns a constraint that ensures that, whenever a decision interval variable $dvar$ is present, a state function f is defined everywhere between the start and the end of the decision interval variable $dvar$.

State Function is a decision variable whose value is a set of non-overlapping intervals over which function f is defined. Between these intervals, f is not defined, usually because of an ongoing transition between two states.

Now the CP model of [XD]:MT-SR-TA-SP can be defined. First, let us define the Decision Interval Variables (DIVs). The tasks to be performed are represented by the union of sets of physical $\mathbb{PT}$ and virtual $\mathbb{VT}$ tasks. The total number of tasks is $n = |\mathbb{PT}| + |\mathbb{VT}|$. We also define the set of all tasks to be $\mathbb{T} = \mathbb{PT} \cup \mathbb{VT}$, where each element $t_i \in \mathbb{T}$ corresponds to a single DIV. Similarly, a source depot is denoted as $\sigma_i \in \Sigma$, and a destination depot is denoted as $\delta_i \in \Delta$. DIVs related to source depots are fixed to allow the mission to start at time 0. The number of agents is equal to the number of source depots, i.e., $m = |\mathbb{S}| = |\Sigma|$. All mentioned DIVs are set as **present**.

Now we can define an allocation matrix $\mathbf{A}_{[n\times m]}$ with rows representing tasks and columns agents. Each element a_{ij} is a decision interval for allocation of task i to agent j. Since a task can be allocated to one and only one agent, a_{ij}, it is set to **optional**. As it is stated previously, not every agent can perform every task. In the case of agent j not having the necessary equipment to perform task i, the decision interval a_{ij} is set to **absent**. Another matrix that we need to define is the allocation matrix $\mathbf{AD}_{[m\times|\Delta|]}$, where each element ad_{ij} is a decision interval for the allocation of the agent i to the destination depot j. An agent can be allocated to only one destination depot, hence, ad_{ij} is set to **optional**.

Transition distances between physical tasks, source, and destination depot are expressed with the matrix $\mathbf{M}$, where ω_{ijs} is the cost of transitioning from state i to state j with agent s. The transition distance matrix $\mathbf{M}$ must satisfy the triangular inequality. Virtual tasks are not part of the transition matrix. 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 agent i. The indices of the matrix $\mathbf{M}$ are numbered according to the sequence $\{1, \ldots, |\Sigma| + |\mathbb{PT}| + |\Delta|\}$, where the first $|\Sigma|$ indices are associated with the source depots, the next $|\mathbb{PT}|$ indices are associated with the physical tasks, and the last $|\Delta|$ indices are associated with the destination depots. For the sake of notation, let us introduce the following subsets of indices, $\mathbb{P}^s := \{1, \ldots, |\Sigma|\}$, $\mathbb{P}^{pt} := \{|\Sigma| + 1, \ldots, |\Sigma| + |\mathbb{PT}|\}$, and finally $\mathbb{P}^d := \{|\Sigma| + |\mathbb{PT}| + 1, \ldots, |\Sigma| + |\mathbb{PT}| + |\Delta|\}$, that are going to be used in the following formulation.

The objective is the same as in the ILP model, i.e., to minimize the makespan of the mission. It is assumed that an agent has finished its mission when

$$\text{minimize} \qquad \max_{\delta_i \in \Delta}(StartOf\{\delta_i\})$$

subject to constraints (14) – (22).

Constraint (14) states that for every task t_i, one and only one agent from the set of available agents $\mathbf{A}_i$ can be selected to perform task t_i. For example, if the decision interval variable a_{23} is set to **present** by the solver, it means that task 2 is allocated to agent 3.

$$Alternative(t_i, \mathbf{A}_i), \forall t_i \in \mathbb{T}. \tag{14}$$

where $\mathbf{A}_i$ indicates the i-th row of the matrix $\mathbf{A}$. Constraint (15) states that every agent ends its tour at one of the destination depots. For example, if the decision interval variable ad_{14} is set to **present** by the solver, it means that agent 1 is ending its tour in destination depot 4.

$$Alternative(\delta_i, \mathbf{AD}_i), \forall \delta_i \in \Delta. \tag{15}$$

Precedence constraints are imposed with Eq. (16). If task t_i precedes task t_j, then task t_i must end before task t_j can start.

$$EndBeforeStart(t_i, t_j), \forall t_i, t_j \in \mathbb{T}, t_i \prec t_j. \tag{16}$$

Enforcing all tasks to be assigned to a specific agent before that agent can reach the destination depot is ensured with Eq. (17).

$$EndBeforeStart(a_{ij}, ad_{jk}), \tag{17}$$

for every $i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}, k \in \{1, \ldots, |\Delta|\}$. Constraint (18) states that tasks cannot run in parallel on individual agents except when it is allowed with the matrix $\mathbf{R}$, i.e., task i and task j can run in parallel only if $r_{ij} = 1$.

$$NoOverlap(a_{ik}, a_{jk}), \tag{18}$$

for every $i, j \in \{1, \ldots, n\}, \forall k \in \{1, \ldots, m\}, r_{ij} \neq 1$. In order to enforce the agents' position at the source depot at the start of their mission, we introduce Eq. (19).

$$AlwaysEqual(f_i, \sigma_i, \mathbb{P}^s_i), \forall i\{1, \ldots, m\}, \tag{19}$$

where $\mathbb{P}^s_i$ is the i-th element in the set $\mathbb{P}^s$. In order to enforce the agents' position at one of the destination depots at the end of their mission, we introduce Eq. (20).

$$AlwaysEqual(f_i, ad_{ij}, \mathbb{P}^d_i), \tag{20}$$

for every $i \in \{1, \ldots, m\}, \forall j \in \{1, \ldots, |\Delta|\}$. Constraint (21) uses a state function to model the positions of agents with respect to physical tasks.

$$AlwaysEqual(f_i, a_{ji}, \mathbb{P}_j^{pt}), \tag{21}$$

for every $i \in \{\forall s \in \mathbb{S} | \phi_c(j) \in \mathbb{C}_s\}, j \in \mathbb{P}^{pt}$, where the set $\{\forall s \in \mathbb{S} | \phi_c(j) \in \mathbb{C}_s\}$ is the set of agents that can execute the physical task j that requires the equipment $\phi_c(j)$, as they have the correct equipment $\mathbb{C}_s$.

In case it is required that the same agent performs both tasks i and j, constraint (22) can be used.

$$PresenceOf(a_{ik}) = PresenceOf(a_{jk}), \tag{22}$$

for every $i, j \in \{1, \ldots, n\}, \forall k \in \{1, \ldots, m\}$. This completes the CP model of the [XD]:MT-SR-TA-SP problem configuration. In the next section, we will present a simple case study to put the defined models into a real-world perspective.

VI. A CASE STUDY

This case study has the goal to showcase how the presented formalization of the [XD]:MT-SR-TA-SP problem configuration is mapped to a real-world scenario. A case study showing a comparison between a mission with parallel and serial task execution is given by Miloradović et al. [14].

Let us consider a mission scenario, in the precision agriculture domain, with 3 autonomous agents, i.e., two Unmanned Aerial Vehicles (UAV 1 and UAV 2) and one Unmanned Ground Vehicle (UGV). In this case study, the UAVs have capabilities of visual inspection (e.g., scanning the crops, or inspecting an animal herd), analyzing gathered data, and communicating the data to other agents in the mission, or a computer located on the ground. The UGV has the capabilities to create prescription maps and apply fertilizer to crops. These three agents have been tasked with a mission consisting of 10 tasks, out of which 5 tasks are virtual, and 5 tasks are physical. For simplicity reasons, let us use letters to mark the locations of physical tasks, i.e., Task 1 has location A, Task 2 has location B, Task 3 has location C, Task 4 has location D, and finally Task 7 has location E. The subset containing virtual tasks is consisting of tasks $\{5, 6, 8, 9, 10\}$. The only tasks that can execute in parallel are tasks $(3, 6)$ and $(2, 8)$, i.e., $r_{36} = r_{63} = 1$, and $r_{28} = r_{82} = 1$.

Two crop fields need to be scanned, Tasks 1 and 4. The data needs to be analyzed and forwarded to the appropriate agent, these are Tasks 5 and 8. Based on that data, the appropriate agent will be tasked with creating prescription maps for both crop fields, tasks 6 and 9. Next, based on the generated prescription maps, the UGV is performing the tasks of spraying the crops with the optimal amount of fertilizer, through Tasks 3 and 7. In addition, a visual inspection of the herd of livestock needs to be done (Task 2), and the gathered data needs to be sent to the command center (Task 10).

The aforementioned tasks have several constraints between them. In the case of tasks (1,5) and (4,8), crop scanning tasks have to be done before data analysis tasks, i.e., $1 \prec 5$ and $4 \prec 8$. In addition, tasks 1 and 5 are constrained to be executed by the same agent. The same applies to tasks 4 and 8. In tasks 6 and 9, prescription maps of the corresponding areas are created. Task 6 has a precedence relation with Task 8, i.e., $8 \prec 6$. This means that gathered data first needs to be sent to an agent capable of creating prescription maps, in this case, that is the UGV. For the same reason, $5 \prec 9$. The precedence relations continue, with $9 \prec 3$ and $6 \prec 7$, where tasks 3 and 7 assume the usage of the UGV for applying fertilizer to the crops. The UGV is capable of performing task 6, the creation of a prescription map for Task 7, simultaneously with spraying crops, which is Task 3. Similarly, UAV 1 is capable of analyzing and sending gathered data, whilst performing task 2, which is the inspection of a herd of livestock. Finally, the data gathered in Task 2 is sent to the command center, where we also have a precedence constraint Task $2 \prec 10$. The information about tasks and their mutual relationship is summed up in Table 1.

TABLE 1. Detailed information about tasks in the mission.

ID	Task Description	Type	PC	Parallel with
T1	Crop field scan	Physical	$1 \prec 5$	-
T2	Visual livestock inspection	Physical	$2 \prec 10$	8
T3	Crops fertilizing	Physical	-	6
T4	Crop field scan	Physical	$4 \prec 8$	-
T5	Analyze/Send data	Virtual	$5 \prec 9$	-
T6	Prescription map creation	Virtual	$6 \prec 7$	3
T7	Crops fertilizing	Physical	-	-
T8	Analyze/Send data	Virtual	$8 \prec 6$	2
T9	Prescription map creation	Virtual	$9 \prec 3$	-
T10	Send livestock data	Virtual	-	-

The aforementioned mission is given to the CP solver and the resulting plan is shown in Fig. 2 as three sub-figures, each representing one agent in the mission. Tasks in Fig. 2 are colored in the following way; green-colored boxes with a dotted outline represent transits between two physical tasks, or between source/destination depot and a physical task; red-colored boxes with a full outline represent physical tasks, and blue-colored boxes with a full outline represent virtual tasks. The numbers inside the boxes stand for task IDs.

The plan for UAV 1 is to go to location D and perform crop scanning task (Task 4). When the UAV 1 is done with scanning crops, it performs analysis and forwards gathered data (Task 8) from crops scanning task while moving towards location B, where it starts (livestock inspection (Task 2) simultaneously with sending data gathered from crops scanning. The next task in the plan is analysis and forwarding of gathered data (Task 10) from livestock inspection (Task 2), which is done while transiting from location B to the destination depot.

The UAV 2 has only two tasks allocated to it. The first task is crop scanning (Task 1) that is performed at location A. After crop scanning is completed, UAV 2 proceeds to the destination depot while performing analysis and forwarding of gathered data (Task 5) from crop scanning (Task 1).

The UGV does not start its mission at 0 time. Since there is a cross-schedule dependency between receiving crops data

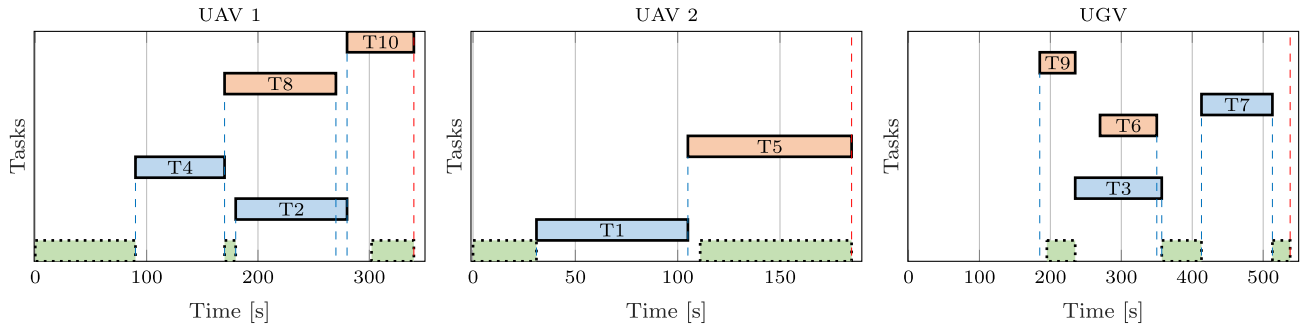

FIGURE 2. The optimized mission plan for the case study with parallel task execution.

(Task 5) and creation of prescription map (Task 9) ($5 \prec 9$), the UGV waits until crops data is received to start with a creation of the prescription map for the crops scanned in Task 1. During the execution of Task 9, the UGV is moving towards location C where the fertilization of the crops should be performed. When the creation of the prescription map is done, spraying the crops (Task 3) starts its execution. During the process of spraying crops by the UGV, the UAV 1 finishes sending data (Task 8), which was a prerequisite for the creation of the prescription map (Task 6) for the crops scanned in Task 4) to start. Tasks 6 and 8 also have cross-schedule dependencies. After the completion of Task 3, the UGV proceeds to its final assignment, another crop field fertilization (Task 7) . Since the prescription map was created in Task 6, crop fertilization can start as soon as the robot reaches location E. Finally, when crop fertilization is done, the UGV proceeds to its destination depot, and the mission is completed.

In this small mission, which can also be seen as a part of a larger mission, we have demonstrated a real-world use case of a [XD]:MT-SR-TA-SP problem configuration.

VII. EVALUATION

While CP is a common approach when solving scheduling operations [22], it is shown that (M)ILP can be competitive in addressing these types of problems [23], and it has been widely used for routing operations. The model presented in this paper can be seen as a mixture of both routing and scheduling. For this reason, we evaluate our proposed ILP and CP models by implementing them in the CPLEX optimization tool and the CP Optimizer, respectively. The version of both tools is v20.1.0. The implemented models are benchmarked on an extensive set of test instances, with gradually increasing complexity, in order to reveal the behavior of the algorithm extensively. The experimental platform is an i9-9980XE @4.1GHz (18 cores) CPU with 128 GB of DDR4 RAM. It is worth noticing that the planning activity is typically carried out offline, before the mission. However, in case of unexpected events a fast, time-limited, re-planning is necessary. Each of the solvers is given 2 hours to find the best solution.

A. BENCHMARK SETUP

The benchmark consists of 10 problem instances with different levels of complexity, with respect to the number of agents, tasks, task types, equipment, and precedence constraints. We limit the number of different equipment required by tasks to 3. The number of required precedence relations is in relation to the number of tasks involved in a mission, i.e., 0% to 35% of the tasks will have precedence relations. The number of parallel tasks is set to 50%, while the number of virtual tasks varies between 10, 50, and 90% of all the tasks in the mission. The number of tasks and agents also varies depending on the test instance. The task duration is randomized in the range of 10 to 500 seconds. The instances are randomly created with the aforementioned limitations. The exact numbers for each test instance are given in Table 2, more specifically, the first row shows the number of agents available in each problem instance, the next row shows the number of tasks to be done, followed by the number of precedence constraints between tasks, the next row shows the number of source depots per problem instance, and finally, the number of destination depots.

Two solvers are compared based on the set of 10 different test instances, which have gradually increasing complexity in terms of the number of agents, tasks, and precedence constraints, to better understand the scalability issues of the solvers. Both solvers are running in deterministic mode, meaning that they would produce the same solution each run. For this reason, each test instance is run only once and the value of this one run is shown in figures and tables in this section. Each instance was created with a different percentage of virtual tasks, more precisely with 10, 50, and 90% of virtual tasks. We created three different benchmarks with different percentages of virtual tasks in order to analyze the effect of virtual tasks on the two solvers. Both solvers are run for the same amount of time (7200 seconds = 2 hours) and we compared the best results found, as well as the gap between the lower bound and the best solution found for every test instance. In addition, we also compare the convergence speed of the two solvers.

B. IMPLEMENTATION

The performance of the solver greatly depends on the way a model is implemented. Although the goal of this

TABLE 2. Detailed information about each test instance.

Instance	1	2	3	4	5	6	7	8	9	10
# Agents	1	1	2	2	3	3	4	4	5	5
# Tasks	5	8	10	12	14	16	18	20	25	30
# Prec.	0	1	1	4	2	1	5	3	5	11
# Src. dep.	1	1	2	2	3	3	4	4	5	5
# Dest. dep.	1	1	2	2	3	3	4	4	5	5

TABLE 3. Number of constraints and variables in CPLEX and CPO.

	CPLEX		CPO	
Instance	Variables	Constraints	Variables	Constraints
1	70	186	24	25
2	156	646	48	53
3	790	2829	130	159
4	850	4357	65	83
5	1755	11259	183	241
6	2442	16402	342	438
7	3672	32546	393	564
8	4304	43127	438	612
9	8813	106969	1188	1622
10	10596	174208	988	1377

TABLE 4. Results for benchmark 1 with 10% of virtual tasks.

	CPLEX			CPO		
Inst.	Cost	GAP	Time [s]	Cost	GAP	Time [s]
1	**862**	**0%**	0.1	**862**	**0%**	**0.006**
2	**1222**	**0%**	1.1	**1222**	**0%**	**0.17**
3	**727**	**0%**	**58.5**	**727**	43%	7200
4	1205	98%	7200	**1198**	**72%**	7200
5	741	100%	7200	**664**	**88%**	7200
6	996	100%	7200	**715**	**53%**	7200
7	724	100%	7200	**488**	**88%**	7200
8	1275	100%	7200	**498**	**89%**	7200
9	/	/	7200	**593**	**94%**	7200
10	/	/	7200	**647**	**90%**	7200

work is not to address the optimization of the implemented model, but rather to evaluate the proposed model, we do incorporate some of the implementation details that were used to configure the solver.

1) CPLEX

Disjunctive constraint (Eq. (11)) is implemented as indicator constraint. The formulations with *big-*$\mathcal{M}$ coefficients may cause numerical instability. The provided indicator constraints avoid this problem altogether; however, this comes at the cost of reduced performance in terms of convergence time. Ku and Beck [15] performed a comparison of disjunctive and indicator constraints on a Job-Shop Scheduling Problem and showed that indicator constraints are up to three times slower than the disjunctive ones. Nevertheless, we used the `IloIfThen` indicator constraints provided by CPLEX, to avoid potential numerical issues, as indicated in the CPLEX user's manual [24]. Finally, the search emphasis was set to balance feasibility and optimality.

2) CP OPTIMIZER

The constraint programming model is implemented as it is presented in Sect. V, since the way the model is presented has been influenced by the CP Optimizer in the first place. We ran the solver with the enabled `Presolve`, and `TimeLimit` set to 7200 seconds, as stated earlier.

TABLE 5. Results for benchmark 2 with 50% of virtual tasks.

	CPLEX			CPO		
Inst.	Cost	GAP	Time [s]	Cost	GAP	Time [s]
1	**506**	**0%**	0.12	**506**	**0%**	**0.003**
2	**1010**	**0%**	2.6	**1010**	**0%**	**0.17**
3	536	**68%**	7200	**535**	80%	7200
4	**577**	**0%**	**515**	**577**	39%	7200
5	**632**	100%	7200	**632**	**60%**	7200
6	745	100%	7200	**497**	**88%**	7200
7	787	100%	7200	**540**	**92%**	7200
8	621	100%	7200	**442**	**85%**	7200
9	/	/	7200	**482**	**90%**	7200
10	/	/	7200	**1459**	**78%**	7200

TABLE 6. Results for benchmark 3 with 90% of virtual tasks.

	CPLEX			CPO		
Inst.	Cost	GAP	Time [s]	Cost	GAP	Time [s]
1	**487**	**0%**	0.16	**487**	**0%**	**0.01**
2	**1121**	**0%**	4.3	**1121**	**0%**	**0.15**
3	**454**	**70%**	7200	**454**	77%	7200
4	**740**	**0%**	**312**	**740**	56%	7200
5	**343**	100%	7200	**343**	**68%**	7200
6	701	100%	7200	**461**	**87%**	7200
7	737	100%	7200	**544**	**92%**	7200
8	576	100%	7200	**414**	**84%**	7200
9	/	/	7200	**465**	**90%**	7200
10	/	/	7200	**1406**	**66%**	7200

`DefaultInferenceLevel` is kept at basic, as changing it to extended yielded worse overall results. The other options assumed their default values.

The number of constraints and decision variables for each problem instance is shown in Table 3 for both CPLEX and CPO. Values for CPLEX are given after the automatic presolve operation, which is used to reduce the number of constraints and decision variables by removing the redundant ones. Eq. (9) is responsible for more than 90% of the total number of constraints in the case of CPLEX. Provided values are for the case when 50% of the tasks in the mission are virtual.

C. RESULTS

The obtained results for the ILP model in CPLEX and the CP model in CPO are given in Tables 4, 5, and 6. Each benchmark consists of 10 test instances. In total, we conducted three different benchmarks, each with a different percentage of virtual tasks. The reported computation times exclude the time taken to generate the model from the ILP/CP formulations.

The first instance is solved by both solvers, with a 0% gap in each of the 3 benchmarks. When the gap is 0% it means that the found solution is guaranteed to be the optimal one. The gap is defined as $\frac{(bs-bb)}{(bs\cdot 10^{-2})}$, where bs is the best solution found, and bb is the best bound. Computation time, however, differs between the solvers, where CPO is faster to reach the optimal solution. The situation is similar for Instance 2, where both solvers reach an optimal solution, but CPO is an order of magnitude faster. Instance 2 is the last instance where CPO was able to prove that the solution found was the optimal one. CPLEX performed better in this regard as it was able to prove

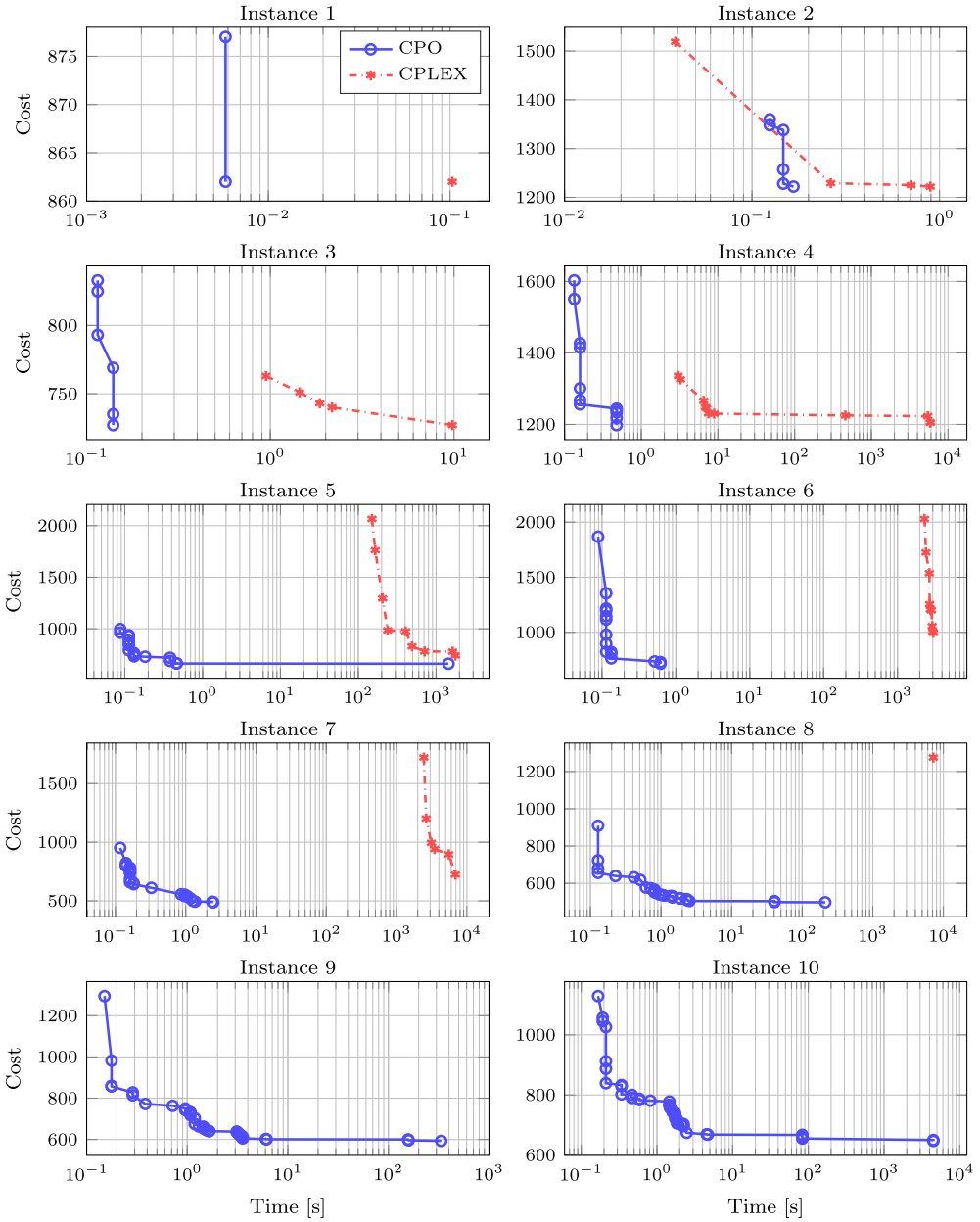

FIGURE 3. Convergence plot for all 10 instances with 10% of virtual tasks.

the optimality of Instance 3 in Benchmark 1, and Instance 4 in benchmarks 2 and 3. In these 3 instances, CPLEX managed to find the optimal solution before the set time limit. In the rest of the instances, in all 3 benchmarks, neither solver managed to prove the optimality of any found solution. Moreover, CPLEX was only able to find a lower bound in Instance 4 of benchmark 1, and Instance 3 of benchmarks 2 and 3. In all other instances, CPLEX was unable to find any lower bound.

In the first benchmark (Table 4), CPLEX and CPO produced the same solutions only for the first three instances. In the next five instances, CPO outperformed CPLEX by 17% on average. In the last two instances, CPLEX was unable to find any feasible solution within the given time limit. CPO was also able to provide a better lower bound for instances 4–10.

In the second benchmark (Table 5), for instances 1-5, both solvers found solutions of the same quality. For the first 4 instances, CPLEX had the advantage in providing a better lower bound, while CPO performed better, both in terms of solution quality and lower bound, as instances got harder. For the last two instances, CPLEX was unable to find any feasible solution within the time limit. In the first 8 instances, CPO outperformed CPLEX by 12% on average.

The third benchmark (Table 6) showed similar results to the second benchmark. Both solvers found the same solution for the first 5 instances, with CPLEX providing a better

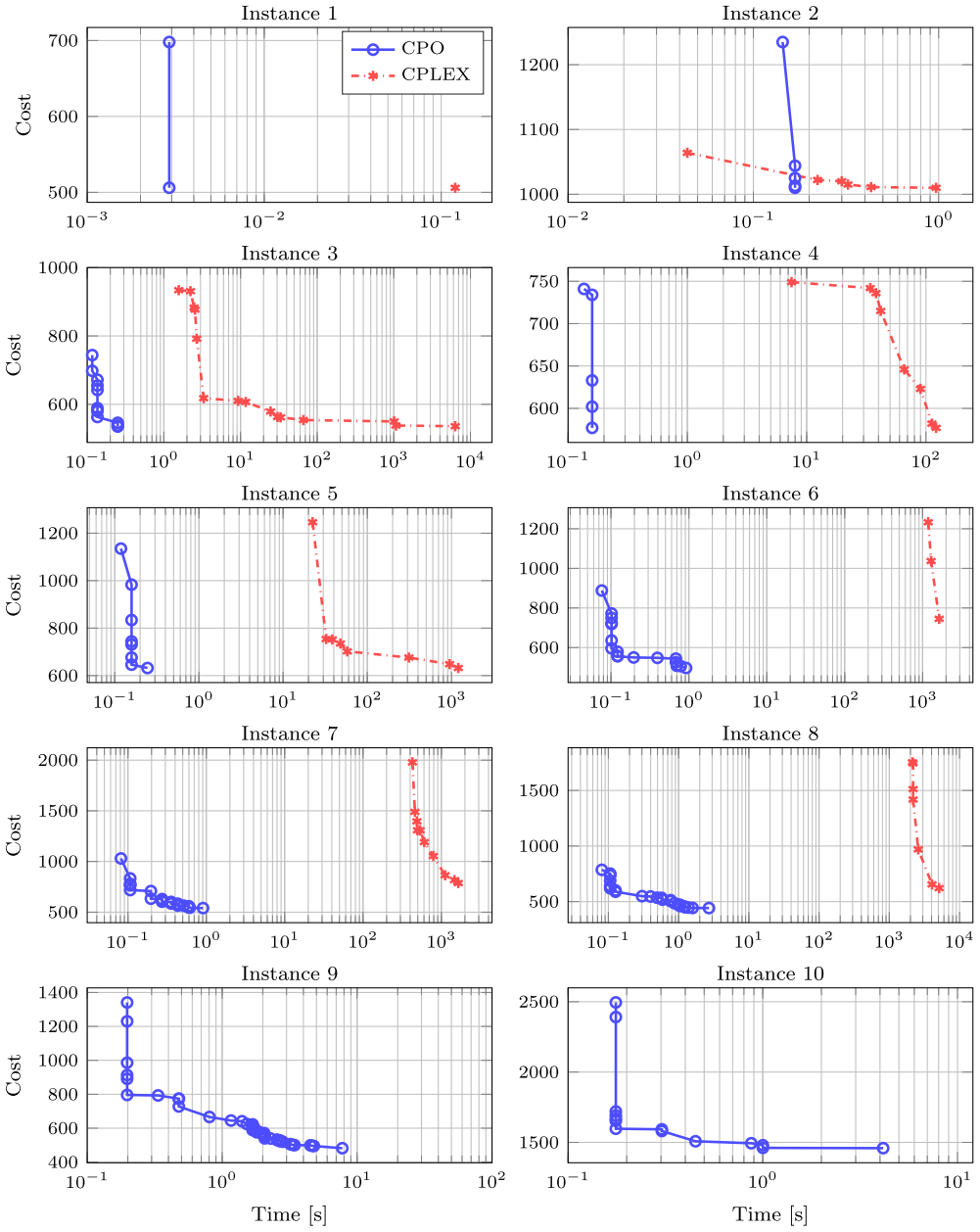

FIGURE 4. Convergence plot for all 10 instances with 50% of virtual tasks.

lower bound in the first 4. CPO performed better with instances getting harder, compared to CPLEX, both in terms of solution quality and lower bound. Although, the gap was significantly large (between 66% and 92%). The last two instances remained unsolved by CPLEX, the same as in other benchmarks, however, the average difference on other instances was only 11% in favor of CPO.

What can be concluded from these benchmarks is that CPLEX is a viable option for small problem instances, where guaranteed optimality is necessary. In every other case, CPO presents itself as a clear winner. Instance 4, in both benchmarks 2 and 3, shows that a mission with more tasks is not necessarily more difficult for a solver since Instance 3 was not solved to optimality, contrary to Instance 4. It is also noticeable that the gap between the solution quality of CPLEX and CPO narrowed with the increase in the number of virtual tasks. Nevertheless, CPO always produced solutions that are at least as good as the output of the ones by CPLEX, and it was able to find feasible solutions where CPLEX has failed to do so. In our benchmarks we had a time limit fixed to 7200 seconds. While this might be a reasonable time for some missions, other missions might require feasible plans in a smaller time span. Particularly in cases where unforeseen events may change the state of the mission and/or the environment. In such a situation, a fast response is necessary in order to ensure the successful completion of

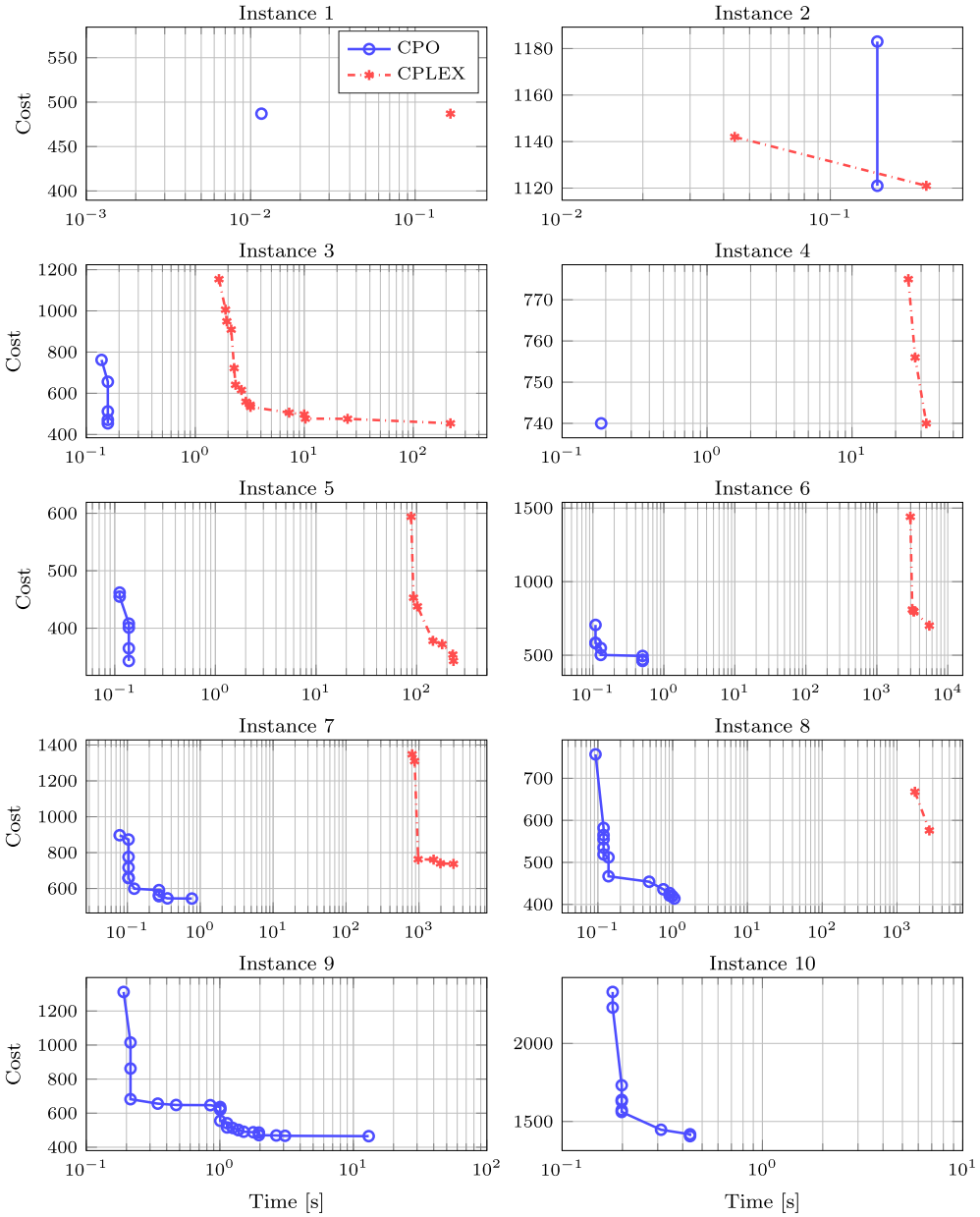

FIGURE 5. Convergence plot for all 10 instances with 90% of virtual tasks.

the initial mission goal. This is why time is critical when it comes to re-planning a mission. In order to determine which algorithm is better in this department, we will take a look at the convergence rate of both CPO and CPLEX.

Figs. 3, 4, and 5 show the convergence rates for every instance in each of the three benchmarks. What can be noticed is that CPO's convergence rates are far superior to CPLEX. In most cases (Instances 1–8, except for instances 5 and 8 with 10% virtual tasks, Fig. 3), CPO converges within the first second. Even for the harder problems, instances 9 and 10, CPO does most of its convergence within the first 10 seconds, especially for benchmarks 2 (Fig. 4) and 3 (Fig. 5), i.e., with 50 and 90% of virtual tasks in a test instance. In most cases (except Instance 2), CPLEX takes longer to find an initial feasible solution, and more often than not, that solution is worse compared to the initial solution found by CPO. From these results, it is clear that CPO is more suitable for situations where mission re-planning is needed and that, on average, CPO is faster to find an initial feasible solution, has a faster convergence rate, and finds better quality solutions compared to CPLEX. CPLEX is competitive only in the first two instances. It is important to emphasize that both CPLEX and CPO have room for improvement. CPLEX has 76 tunable parameters, as stated by Hutter et al. [25] in their paper on automated configurations for MIP solvers. They managed to improve proving optimality by the speedup factor of up to 52 and to improve gap minimization with a factor of up to 42, on some MIP problems. Similarly, CPO has a huge

number of tunable parameters [26], which can greatly impact the solver's performance. We used default settings wherever possible, since parameter tuning is a time-consuming process on its own, so it should be used only when there is a clear benefit.

It is can be concluded that both of the proposed approaches have limitations. CPLEX has the advantage of being able to provide tighter guarantees on the quality of the solution found, however, it can be very slow in finding good or even feasible solutions. CPO on the other hand is much faster, as Tables 4, 5, 6 and Figs. 3, 4, 5 show, but the guarantees on the quality of the solution are worse compared to CPLEX.

VIII. RELATED WORK

As stated in Sect. III, there are not many papers on the problem configuration described in this work. Specifically, approaches that incorporate Multi-Task Robot systems seem to be missing from the literature, at least in terms of how it is defined in the original MRTA taxonomy. That being said, there are a certain number of papers addressing multitask robots dimension, e.g., [27]. However, the term multitask robots, in their work, refers to robots that can do multiple different tasks, but not simultaneously. Due to the absence of literature on the problem described in this paper, we will address some problems that are similar to [XD]:MT-SR-TA:SP.

Lacomme et al. [28] have presented a model of the resource-constrained project scheduling problem with routing. This is a closely related problem, but the vehicles used in Lacomme's work can be categorized as Single-Task Robots. Cordeau and Laporte [29] have presented the dial-a-ride VRP problem. Depending on the level of abstraction used for tasks, one can argue that this type of problem falls under the MT-SR-TA category. However, we do not consider this to be the case based on the discussion in Sect. II. Nevertheless, this work is closely related to the work presented in this paper. The subject of MT robots with an emphasis on complex task allocation has been addressed by Landèn et al. [30]. Although the authors mention that their problem can be classified as MT, it is unclear if it really is. By the definition of [3], MT robots perform tasks simultaneously. The example Landen et al. provided, refers to scanning two different areas, A and B, as an MT. This cannot be done simultaneously by a single agent. Cano et al. [31] have presented a paper where they perform the allocation of software tasks to CPUs. The problem is modeled as a multi-objective, multidimensional, multiple-choice knapsack problem, that falls under the MRTA category of XD:SR-MT-IA. Each task is a set of task variants that have different parameters for Quality of Service and Utilization. Finally, if we consider the ST-MR-TA problem configuration, which is in a similar problem domain as the work presented here, Sariel and Balch [32] worked on the allocation of resource-constrained project tasks to robots. The work handles the dynamic allocation of tasks with precedence and synchronization constraints. In addition to PCs, tasks also require a certain number of robots to execute them.

IX. CONCLUSION

When a MAS instance consists of a set of very diverse autonomous agents having extensive computation and manipulation capabilities, it is necessary to provide a model that can exploit all possible solutions through appropriate choices for the agents and task pairs. In this work, we have presented a mathematical model for the optimization of multi-robot mission planning, which can exploit task parallelism. Furthermore, we have introduced the distinction between virtual and physical tasks, characterizing their relations in terms of parallel task execution. The proposed formulation utilizes the proposed task separation in an attempt to minimize the overall makespan of the mission. In particular, we have focused on the formulation of the [XD]:MT-SR-TA-SP problem configuration, both as ILP and CP problem, implemented in CPLEX and CPO, respectively. The mission goal was the minimization of the longest agent's makespan over all the agents. The presented models have been evaluated in 3 benchmarks, each consisting of 10 test instances. The results of a comparison of CPLEX and CPO have shown that CPO is far more suitable for this type of problem. It performed at least equally well, or better, in terms of the quality of solutions found. It was also able to find better lower bounds for harder instances. The only advantage CPLEX had is managing to prove optimality for a limited number of small instances. These results also exposed CPLEX's scalability issue with these types of problems. Future work will extend this formulation to include Time-Windows and Multi-Robot tasks. Additionally, further investigation is needed to test CP Optimizer on larger instances and compare its performance and scalability to a solver that is not a general-purpose tool but specifically tailored for this type of problem.

REFERENCES

[1] M. Frasheri, J. Cano-Garcia, E. Gonzalez-Parada, B. Çürüklü, M. Ekström, A. V. Papadopoulos, and C. Urdiales, "Adaptive autonomy in wireless sensor networks," in *Proc. AAMAS*, 2020, pp. 375–383.

[2] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *IEEE Access*, vol. 6, pp. 28573–28593, 2018.

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

[4] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *Int. J. Robot. Res.*, vol. 32, no. 12, pp. 1495–1512, 2013.

[5] E. Nunes, M. Manner, H. Mitiche, and M. Gini, "A taxonomy for task allocation problems with temporal and ordering constraints," *Robot., Auto. Syst.*, vol. 90, pp. 55–70, Apr. 2016.

[6] F. C. M. J. M. van Delft, G. Ipolitti, D. V. Nicolau, A. Sudalaiyadum Perumal, O. Kašpar, S. Kheireddine, S. Wachsmann-Hogiu, and D. V. Nicolau, "Something has to give: Scaling combinatorial computing by biological agents exploring physical networks encoding NP-complete problems," *Interface Focus*, vol. 8, no. 6, Dec. 2018, Art. no. 20180034.

[7] J. Chen, Y. Zhang, L. Wu, T. You, and X. Ning, "An adaptive clustering-based algorithm for automatic path planning of heterogeneous UAVs," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 16842–16853, Sep. 2022.

[8] J. Chen, C. Du, Y. Zhang, P. Han, and W. Wei, "A clustering-based coverage path planning method for autonomous heterogeneous UAVs," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 25546–25556, Dec. 2022.

[9] J. Chen, F. Ling, Y. Zhang, T. You, Y. Liu, and X. Du, "Coverage path planning of heterogeneous unmanned aerial vehicles based on ant colony system," *Swarm Evol. Comput.*, vol. 69, Mar. 2022, Art. no. 101005.
[10] P. Toth and D. Vigo, "The vehicle routing problem," in *Discrete Mathematics and Applications*. Philadelphia, PA, USA: SIAM, 2002.
[11] K. Jansen, M. Mastrolilli, and R. Solis-Oba, "Approximation schemes for job shop scheduling problems with controllable processing times," *Eur. J. Oper. Res.*, vol. 167, no. 2, pp. 297–319, Dec. 2005.
[12] S. Seok, D. J. Hyun, S. Park, D. Otten, and S. Kim, "A highly parallelized control system platform architecture using multicore CPU and FPGA for multi-DoF robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 5414–5419.
[13] J. Vander Hook, T. Vaquero, F. Rossi, M. Troesch, M. S. Net, J. Schoolcraft, J.-P. de la Croix, and S. Chien, "Mars on-site shared analytics information and computing," in *Proc. Int. Conf. Automated Planning Scheduling*, vol. 29, 2019, pp. 707–715.
[14] B. Miloradovic, B. Curuklu, M. Ekstrom, and A. V. Papadopoulos, "Exploiting parallelism in multi-task robot allocation problems," in *Proc. IEEE Int. Conf. Auto. Robot Syst. Competitions (ICARSC)*, Apr. 2021, pp. 197–202.
[15] W.-Y. Ku and J. C. Beck, "Mixed integer programming models for job shop scheduling: A computational analysis," *Comput. Oper. Res.*, vol. 73, pp. 165–173, Sep. 2016.
[16] B. Miloradović, B. Curuklu, M. Ekström, and A. V. Papadopoulos, "TAMER: Task allocation in multi-robot systems through an entity-relationship model," in *Proc. Int. Conf. Princ. Pract. Multi-Agent Syst.* Cham, Switzerland: Springer, 2019, pp. 478–486.
[17] J. Li, M. Zhou, Q. Sun, X. Dai, and X. Yu, "Colored traveling salesman problem," *IEEE Trans. Cybern.*, vol. 45, no. 11, pp. 2390–2401, Nov. 2015.
[18] B. Miloradovic, B. Çürüklü, M. Ekström, and A. V. Papadopoulos, "GMP: A genetic mission planner for heterogeneous multirobot system applications," *IEEE Trans. Cybern.*, vol. 52, no. 10, pp. 10627–10638, Oct. 2022.
[19] M. Gini, "Multi-robot allocation of tasks with temporal and ordering constraints," in *Proc. 31st AAAI Conf. Artif. Intell. (AAAI)*, 2017, pp. 1–7.
[20] P. Laborie and J. Rogerie, "Reasoning with conditional time-intervals," in *Proc. 21st Int. FLAIRS Conf.*, 2008, pp. 555–560.
[21] P. Laborie, J. Rogerie, P. Shaw, and P. Vilím, "Reasoning with conditional time-intervals. Part II: An algebraical model for resources," in *Proc. 22nd Int. FLAIRS Conf.*, 2009, pp. 201–206.
[22] R. Barták, M. A. Salido, and F. Rossi, "New trends in constraint satisfaction, planning, and scheduling: A survey," *Knowl. Eng. Rev.*, vol. 25, no. 3, pp. 249–279, Sep. 2010.
[23] S. Heinz and J. C. Beck, "Reconsidering mixed integer programming and MIP-based hybrids for scheduling," in *Proc. Int. Conf. Integr. Artif. Intell. (AI) Oper. Res. (OR) Techn. Constraint Program.* Berlin, Germany: Springer-Verlag, 2012, pp. 211–227.
[24] IBM. (2017). *IBM ILOG CPLEX Optimization Studio CPLEX User's Manual*. IBM. [Online]. Available: https://www.ibm.com/support/knowledgecenter/SSSA5P_12.8.0/ilog.odms.studio.help/pdf/usrcplex.pdf
[25] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Automated configuration of mixed integer programming solvers," in *Proc. Int. Conf. Integr. Artif. Intell. (AI) Oper. Res. (OR) Techn. Constraint Program.* Berlin, Germany: Springer-Verlag, 2010, pp. 186–202.
[26] P. Laborie, J. Rogerie, P. Shaw, and P. Vilím, "IBM ILOG CP optimizer for scheduling," *Constraints*, vol. 23, no. 2, pp. 210–250, 2018.
[27] K. Ishii, K. Takasuna, and M. Imai, "Collaborative task casting for multi-task communication robots," in *Proc. 16th IEEE Int. Symp. Robot Human Interact. Commun. (RO-MAN)*, Aug. 2007, pp. 338–343.
[28] P. Lacomme, A. Moukrim, A. Quilliot, and M. Vinot, "Integration of routing into a resource-constrained project scheduling problem," *EURO J. Comput. Optim.*, vol. 7, no. 4, pp. 421–464, Dec. 2019.
[29] J.-F. Cordeau and G. Laporte, "The dial-a-ride problem: Models and algorithms," *Ann. Oper. Res.*, vol. 153, pp. 29–46, Sep. 2007, doi: 10.1007/s10479-007-0170-8.
[30] D. Landén, F. Heintz, and P. Doherty, "Complex task allocation in mixed-initiative delegation: A UAV case study," in *Proc. Int. Conf. Princ. Pract. Multi-Agent Syst.* Berlin, Germany: Springer-Verlag, 2010, pp. 288–303.
[31] J. Cano, D. R. White, A. Bordallo, C. McCreesh, A. L. Michala, J. Singer, and V. Nagarajan, "Solving the task variant allocation problem in distributed robotics," *Auto. Robots*, vol. 42, no. 7, pp. 1477–1495, Oct. 2018.
[32] S. Sariel and T. Balch, "Dynamic and distributed allocation of resource constrained project tasks to robots," in *Proc. Multi-Agent Robotic Syst. (MARS) Workshop 3rd Int. Conf. Informat. Control, Autom. Robot.*, 2006, pp. 1–10.

BRANKO MILORADOVIĆ (Student Member, IEEE) received the B.Sc. and M.Sc. degrees in mechanical engineering from the University of Belgrade, Serbia, in 2008 and 2010, respectively, and the Ph.D. degree in computer science from Mälardalen University, Sweden, in 2022. His research interests include robotics, multi-robot mission planning, combinatorial optimization, and evolutionary algorithms.

BARAN ÇÜRÜKLÜ received the M.Sc. and Ph.D. degrees in computer science from Mälardalen University, Västerås, Sweden, in 1998 and 2005, respectively. He is currently a Senior Lecturer of artificial intelligence with Mälardalen University. His current research interests include knowledge representation in cortical structures, collaboration in heterogeneous teams of artificial and human agents, and evolution in complex systems.

MIKAEL EKSTRÖM (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in physics from Uppsala University, Sweden, in 1993 and 1999, respectively. He is currently a Professor of robotics with Mälardalen University, Västerås, Sweden, where he is the Head of the Robotics Research Group. His research interests include robotics, autonomous vehicles, sensors, and communication.

ALESSANDRO VITTORIO PAPADOPOULOS (Senior Member, IEEE) received the B.Sc. and M.Sc. (summa cum laude) degrees in computer engineering and the Ph.D. degree (Hons.) in information technology from Politecnico di Milano, Milan, Italy, in 2013. He is currently a Professor of electrical and computer engineering with Mälardalen University, Västerås, Sweden. His research interests include robotics, control theory, real-time systems, and autonomic computing.