Exploiting Parallelism in Multi-Task Robot Allocation Problems

Branko Miloradović
Mälardalen University
Västerås, Sweden
branko.miloradovic@mdh.se

Baran Çürüklü
Mälardalen University
Västerås, Sweden
baran.curuklu@mdh.se

Mikael Ekström
Mälardalen University
Västerås, Sweden
mikael.ekstrom@mdh.se

Alessandro V. Papadopoulos
Mälardalen University
Västerås, Sweden
alessandro.papadopoulos@mdh.se

***Abstract*—Multi-Agent Systems (MASs) have been widely adopted in robotics, as a means to solve complex missions by subdividing them into smaller tasks. In such a context, Multi-Robot Task Allocation (MRTA) has been a relevant research area, with the main aim of providing formulations and solutions to different mission configurations, in order to optimize the planning and the execution of complex missions utilizing multiple robots.**

In recent years, robotic systems have become more powerful thanks to the adoption of novel computing platforms, enabling an increased level of parallelism, in terms of sensing, actuation, and computation. As a result, more complex missions can be achieved, at the cost of an increased complexity for the optimization of the mission planning.

In this paper, we first introduce the distinction between physical and virtual tasks of the robots, and their relation in terms of parallel execution. Therefore, we propose a mathematical formalization of the mission planning problem for Multi-Task (MT) robots, in the presence of tasks that require only a Single-Robot (SR) to complete, and in the presence of Time-Extended Assignments (TAs). The problem is modeled with a Mixed-Integer Linear Programming (MILP) formulation, with the objective of minimizing the total makespan of the mission, exploiting the potential (physical and virtual) parallelism of the robots. The model is validated over some representative scenarios, and their respective solutions are obtained with the CPLEX optimization tool, showing the generality of the proposed formulation.

***Index Terms*—Multi-Robot Task Allocation, Parallel Task Execution, Mixed-Integer Linear Programming**

I. Introduction

Multi-Agent Systems (MASs) have been widely adopted in robotic systems for the coordination, cooperation, and planning of multiple robots [1]–[3]. More specifically, complex robotics missions can be subdivided into smaller tasks, to better exploit the distributed nature of MAS as well as the potential heterogeneity of the considered agents. The resulting mapping between the tasks composing the mission and the robots is known as the Multi-Robot Task Allocation (MRTA) problem [4], and it has been an active research area of the past two decades. The richness of the problem is witnessed by the numerous taxonomies proposed in the literature [4]–[6].

A multi-agent mission involves a number of tasks, a number of agents, and a set of constraints describing the mutual relations of tasks and agents. A multi-agent mission planning can be then defined as the allocation of a set of tasks to a set of agents with respect to agent capabilities and task requirements, and it is equivalent to the MRTA problem. Although it is possible to let a human to manually plan a mission for very small instances, solving the MRTA problem efficiently requires automated planning algorithms to be able to deal with high number of tasks and constraints; however, even automated planning falls short in terms of scalability [7]. If a mission consists of a large group of agents that can perform multiple mutually interrelated tasks in parallel, which are also scattered in the environment, the problem can be described as a mixture of routing and scheduling problems. Even in their simple forms, both routing [8] and scheduling [9] problems are NP-hard. Consequently, the combination of these two problems is also at least NP-hard.

In recent years, robotic systems computational capacities have become more powerful, thanks to the advances in modern parallel real-time computing technologies [10], enabling an unprecedented level of parallelism, in terms of sensing, actuation, and computation. As a result, more complex missions can be achieved by the robotic system, including tasks that require only computational capacity, and no physical actuation, e.g., data processing or data transmission [11]. The additional tasks to be assigned to the robot come at the cost of an increased complexity for the MRTA and, more in general, for the mission planning. In this paper, we introduce a new distinction between physical and virtual tasks, and their relation in terms of parallel execution. This distinction captures the additional computational capabilities of robotic systems, and it allows for a more rich specification of the task set required to complete the mission.

The contribution of this paper is twofold: (i) we propose a mathematical model, in the form of Mixed-Integer Linear Programming (MILP) problem, for the optimization of the makespan of the problem configuration that covers Time-extended Assignment (TA) of Single-Robot (SR) tasks, both physical and virtual tasks, to Multi-Task (MT) robots. The model is verified with the CPLEX solver on a set of problem instances; (ii) we propose 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.

This work was supported by the project Aggregate Farming in the Cloud (AFarCloud) European project, with project number 783221 (Call: H2020-ECSEL-2017-2), by the Knowledge Foundation with the FIESTA project, and by the Swedish Research Council (VR).

II. Background and Related Work

The original MRTA taxonomy [4] for problem classification, covers three main problem dimensions resulting in a total of 8 different problem configurations. As the authors stated at the time, 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, the authors state that the MT-SR-TA configuration can be seen as a ST-MR-TA (MR stands for Multi-Robot tasks) with the set partitioning problem inverted, i.e., splitting a set of tasks into agent-specific coalitions. Such a formulation does not take into account the location of the tasks and the time required for the robots to move between them, focusing only on the task allocation problem. The interrelatedness of the tasks has never been addressed in [4], which makes the definition of MT or MR configurations incomplete.

The next contribution to the MRTA taxonomy was *iTax* [5], which aims at addressing the task interrelatedness that was missing from the original MRTA taxonomy. This work provides a survey on different problem configurations. The MT configuration lacked mathematical models of formal problem definition, and in general, this topic has not been investigated extensively by the research community. The authors stated that some variants of Vehicle Routing Problem (VRP), like pick-up and delivery, can be seen as an example of MT problem configuration. Assuming that from the pick-up time until delivery we have a continuously ongoing task, then every stop to pick or deliver other packages can be seen as a from of parallel task execution. The problem here is the level of abstraction of tasks. If we assume that the task starts when the package is picked up and ends when that package is delivered, VRP can be considered to have MT-SR-TA configuration. However, this description assumes a high-level description, since the task composes 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 in 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 in a preemptive manner. For example, picking up of 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 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.

From the *iTax* we can conclude that if tasks have relations to other tasks within the same agent's schedule, the problem falls into 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, 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.

These taxonomies have been extended with two new dimensions, a Synchronization Precedence (SP) and Time Windows (TW) [6]. The former refers to an ordering constraints, e.g., task A needs to be done before task B, either by the same agent or different agent; whereas 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 SP dimension. In addition, the authors 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 taxonomy extension [12], the MT part has not been further explored, but it was kept as in the original MRTA taxonomy. As can be seen form the analysis of the most influential MRTA taxonomies, MT has been completely neglected as a research direction, both from a theoretical and practical perspective [4]–[6], [12]. This is still the case today.

III. Virtual and Physical Task Types

First, we distinguish between two types of tasks that are denoted as *virtual* and *physical* tasks.

Virtual tasks are 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, and so on. 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 during the transition from one task to the next one.

On the other hand, physical tasks are bound to a certain location where they must be performed. Opposite to the 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 (scanning seabed, taking photos of objects of interest, etc.), or using tool (drilling, welding, etc.). Contrarily to what has been proposed by [6], we assume taking pictures of objects as physical task as well, since it has to be done at a specific location. However, taking photos or using 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. It is important to emphasize that in this work we do not deal with preemptive tasks. The parallelism among tasks can be divided into these 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, thus 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 independent tasks and these tasks can be modeled as a single monolithic task.

The situation with the other two cases is however different. Virtual parallelism cannot always be achieved, for example due to functional dependencies among the tasks, to 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 be not 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 underwater interference.

IV. Problem Formulation

Let $s \in \mathcal{S}$ be an agent, in a set $\mathcal{S} := \{s_1, s_2, \ldots, s_m\}$ of m agents that need to perform a set of n tasks, $\mathcal{V} := \{v_1, v_2, \ldots, v_n\}$; $\mathcal{V}$ includes both physical and virtual tasks. Also, let c be a type of equipment in a set $\mathcal{C} := \{c_1, c_2, \ldots, c_k\}$; each agent is endowed with one or more pieces of equipment, and every task requires one specific piece equipment to be completed. We denote with σ a deployment location in a set $\Sigma := \{\sigma_1, \ldots, \sigma_q\}$ of q deployment locations, and δ to be a destination location in a set $\Delta := \{\delta_1, \ldots, \delta_w\}$.

Each agent s starts from a deployment location σ and finishes its tour at a destination location δ. The superset containing all the tasks and the deployment and destination locations is defined as $\widetilde{\mathcal{V}} := \mathcal{V} \cup \Sigma \cup \Delta$. In addition, for convenience, a superset containing all the deployment locations and tasks elements is defined as $\mathcal{V}^{\Sigma} := \mathcal{V} \cup \Sigma$. Analogously, a superset containing all the destination locations and tasks is defined as $\mathcal{V}^{\Delta} := \mathcal{V} \cup \Delta$. Every edge $e(i,j)$ connecting two tasks $i, j \in \widetilde{\mathcal{V}}$, 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 four decision variables x_{ijs}, t_i, z_{ijs}, and $\mathcal{Q}$.

$$x_{ijs} = \begin{cases} 1, \text{ if } s \in \mathcal{S} \text{ starts task } i \in \mathcal{V}^{\Sigma} \text{ before task } j \in \mathcal{V}^{\Delta}, \\ 0, \text{ otherwise.} \end{cases}$$

$$z_{ijs} = \begin{cases} 1, & \text{if } t_i \leq t_j, \\ 0, & \text{otherwise.} \end{cases}$$

The decision variable $x_{ijs} \in \{0,1\}$ defines if agent s travels from task i to task j. The decision variable $z_{ijs} \in \{0,1\}$defines if start time of task i is before or after task j. The decision variable $t_i \in \mathbb{R}_0^+$ defines the starting time of a task i. Finally, the decision variable $\mathcal{Q} \in \mathbb{R}$ is used to express the upper bound of the longest tour performed by an agent. Every task $i \in \mathcal{V}^{\Sigma}$ has a duration $\xi_{is} \in \mathbb{R}_0^+$, representing the amount of time agent s needs in order to complete task i; if $i \in \Sigma \cup \Delta$, $\xi_{is} = 0$. Precedence relations among tasks are described by the adjacency matrix $\Pi = [\pi_{ij}]_{n \times n}$, where $\pi_{ij} = 1$ if $i \prec j$, and 0 otherwise. In addition, every task $i \in \mathcal{V}$ requires a certain equipment $f_c(i)$ for its successful completion, with $f_c : \mathcal{V} \mapsto \mathcal{C}$. Each agent $s \in \mathcal{S}$ has a set of available equipment $\mathcal{C}_s \subseteq \mathcal{C}$. An equipment matrix defines which tasks can be performed by agent s, and it is defined as $\mathcal{A}_s := [a_{ijs}]_{n \times n}$, with

$$a_{ijs} = \begin{cases} 1, & f_c(i) \in \mathcal{C}_s \wedge f_c(j) \in \mathcal{C}_s \wedge \pi_{ij} = 1 \\ 0, & \text{otherwise.} \end{cases}$$

The definition of the equipment matrix $\mathcal{A}_s$ can be extended to include the deployment and destination locations as:

$$\overline{a}_{ijs} = \begin{cases} a_{ijs}, & i, j \in \mathcal{V}, \\ 1, & (i \in \Sigma, j \in \mathcal{V}^{\Delta}) \vee (i \in \mathcal{V}^{\Sigma}, j \in \Delta), \\ 0, & (i, j \in \Sigma) \vee (i, j \in \Delta). \end{cases}$$

As previously explained, tasks are divided into virtual and physical tasks. We can define a function $h : \mathcal{V} \mapsto \{0,1\}$, where $h_i = 1$ if task i is virtual, and 0 otherwise.

We finally define the symmetrical matrix $\mathcal{R} = [r_{ij}]_{n \times n}$ to specify which tasks can be performed in parallel, where $r_{ij} = 1$ if tasks i and j can be done in parallel, and 0 otherwise.

This problem can be seen as a combination of a routing and scheduling problem.

A. Routing Part of the Formulation

In the following, we present the constraints related to the routing part of the problem.

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

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

$$\sum_{s \in \mathcal{S}} \sum_{j \in \mathcal{V}^{\Delta}} x_{ijs} = 1, \qquad \forall i \in \mathcal{V}. \tag{3}$$

$$\sum_{i \in \mathcal{V}^{\Sigma}} x_{ijs} = \sum_{k \in \mathcal{V}^{\Delta}} x_{jks}, \qquad \forall j \in \mathcal{V}, \forall s \in \mathcal{S}. \tag{4}$$

$$\sum_{i \in \Sigma} \sum_{j \in \mathcal{V}^{\Delta}} x_{ijs} = 1, \qquad \forall s \in \mathcal{S}, \tag{5}$$

$$\sum_{i \in \mathcal{V}^{\Sigma}} \sum_{j \in \Delta} x_{ijs} = 1, \qquad \forall s \in \mathcal{S}, \tag{6}$$

$$\sum_{s \in \mathcal{S}} \sum_{j \in \mathcal{V}^{\Delta}} x_{ijs} = \mathcal{B}_i, \quad \forall i \in \Sigma, \tag{7}$$

$$\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} x_{ijs} \leq |\mathcal{N}| - 1, \ \forall \mathcal{N} \subseteq \mathcal{V}, \mathcal{N} \neq \emptyset, \forall s \in \mathcal{S}. \tag{8}$$

It is forbidden for an agent s to perform tasks with incorrect equipment, Eq. (1), defined in the equipment matrix $\mathcal{A}_s$. Exactly one agent $s \in \mathcal{S}$ can start and finish each task, and it can do it exactly once, Eqs. (2) and (3). The same agent that started a task has to finish it, Eq. (4). The start of every tour has to be at the deployment location Eq. (5), while the final destination must always be at one of the destination locations Eq. (6). Note that some agents can go directly from a deployment location to a destination location 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. The number of agents $\mathcal{B}_{\sigma}$ in each

deployment location $\sigma \in \Sigma$ is assumed to be given as an input to the problem, and it is such that $\sum_{i\in\Sigma} \mathcal{B}_i = |\mathcal{S}|$, Eq. (7). Every agent must therefore leave the deployment location to reach either a task or directly a destination location. In order to eliminate sub-tours, the Dantzig-Fulkerson-Johnson (DFJ) sub-tour elimination formulation [13] has been extended and adapted to be applied to this type of problem as. It states that for each nonempty subset $\mathcal{N} \subseteq \mathcal{V}$, the number of edges between the nodes of $\mathcal{N}$ must be at most $|\mathcal{N}| - 1$, Eq. (8). A task i cannot be repeated: $x_{iis} = 0,\ \forall i \in \tilde{\mathcal{V}}, \forall s \in \mathcal{S}$.

B. Scheduling Part of the Problem

In the following, we present the constraints related to the scheduling part of the problem.

$$x_{ijs} \leq z_{ijs}, \quad \forall i \in \mathcal{V}^{\Sigma}, \forall j \in \mathcal{V}^{\Delta}, \forall s \in \mathcal{S}, \tag{9}$$

$$\sum_{s\in\mathcal{S}} (z_{ijs} + z_{jis}) \leq 1, \qquad \forall i, j \in \mathcal{V}. \tag{10}$$

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

$$t_i + \xi_{is} + \omega_{ijs} \cdot \sum_{s\in\mathcal{S}} z_{ijs} \leq t_j,\ \forall i \in \mathcal{V}^{\Sigma}, \forall j \in \tilde{\mathcal{V}}, i \prec j, \tag{12}$$

$$t_j + \mathcal{M} \cdot (1 - \sum_{s\in\mathcal{S}} z_{ijs}) \geq t_i + \mathcal{P}, \quad \forall i \in \mathcal{V}^{\Sigma}, \forall j \in \tilde{\mathcal{V}}, \tag{13}$$

$$t_i \leq \mathcal{Q}, \quad \forall i \in \Delta. \tag{14}$$

$$\sum_{j\in\mathcal{V}} h_j \cdot x_{ijs} = 0, \qquad \forall i \in \Sigma, \forall s \in \mathcal{S}. \tag{15}$$

The scheduling part deals with constraints including the decision variable z_{ijs} that indicates if task i is started before task j by agent s. The relation between x_{ijs} and z_{ijs} is expressed in Eq. (9), meaning that if agent s executes task i right before j ($x_{ijs} = 1$), then $z_{ijs} = 1$, and that if task i is not executed before j ($z_{ijs} = 0$), then also $x_{ijs} = 0$. We introduce Eq. (10) to prevent possible cycles in variable z. In order to maintain the transitive property Eq. (11) is introduced. It ensures that if a task i starts before task k, and task k starts before task j, then task i must start before task j. In the model, we also include precedence constraints between tasks, i.e., a task i must be completed before task j, Eq. (12). 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 task i (t_i), (ii) the duration of task i (ξ_{is}), and (iii) the cost of moving from i to j (ω_{ijs}). The sum of z_{ijs} over all agents is equal to 0 in 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 is equal to 1 it indicates that an agent s performs both tasks i and j. The disjunctive constraint Eq. (13), in conjunction with Eq. (10), ensure that no two tasks can overlap on the same agent, unless it is allowed by their parallelism, expressed by matrix $\mathcal{R}$. $\mathcal{P}$ is a non-negative real number that expresses the time difference between the starting time t_i and starting time t_j. The way $\mathcal{P}$ is calculated is described in the Sect. IV-D. The total duration of the tours of every agent is upper bounded by the decision variable $\mathcal{Q}$ as defined in Eq. (14). To prevent execution of virtual tasks at the beginning of the plan we introduce Eq. (15).

C. Optimization Problem

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

$$\underset{x,z,t,\mathcal{Q}}{\text{minimize}} \qquad \mathcal{Q}$$

subject to Constraints (1)–(15). A mission can have different objective functions [14]. We chose to minimize the maximum makespan of an agent over all agents.

D. Calculating the value of $\mathcal{P}$

The value $\mathcal{P}$ is calculated based on 3 different cases. In general, $\mathcal{P}$ can be described as the time interval between the starting time of task i and the starting time of task j.

1) Case 1: The first case deals with situations when at least one involved task is virtual, while the other can be either virtual or physical, and the execution of the two tasks cannot be in parallel, i.e., $r_{ij} = r_{ji} = 0$. First, the situation where task j is executed after task i is addressed. In this case, the minimum difference between the two starting times is the duration of task i, i.e., $\mathcal{P}_1 = \xi_{is}$. Analogously, when task j is executed before task i, $\mathcal{P}_1$ is equal to the duration of task j, i.e., $\mathcal{P}_1 = \xi_{js}$. This can be further generalized as

$$\mathcal{P}_1 = \xi_{is} \cdot \sum_{s\in\mathcal{S}} z_{ijs} + \xi_{js} \cdot (1 - \sum_{s\in\mathcal{S}} z_{ijs}), \quad i \in \mathcal{V}^{\Sigma}, j \in \mathcal{V}^{\Delta}. \tag{16}$$

2) Case 2: The second case deals with the situation when one task is virtual, and the other is either virtual or physical, and the two tasks can run in parallel, i.e., $r_{ij} = 1$. This allows task j to have the same starting time as task i, thus in both sub-cases, $z_{ijs} = 1$ and $z_{ijs} = 0$, both tasks have the same starting time. Consequently, the value of $\mathcal{P}_2$ is the same. If we expand Eq. (16) to include task parallelism by multiplying $\mathcal{P}_1$ with $(r_{ij} - 1)$ it results in

$$\mathcal{P}_2 = \mathcal{P}_1 \cdot (r_{ij} - 1) = 0, \quad i \in \mathcal{V}^{\Sigma}, j \in \mathcal{V}^{\Delta}, s \in \mathcal{S}. \tag{17}$$

3) Case 3: The last case considers two physical tasks , thus, by the problem description in Sect. III, it is not possible to execute them in parallel, i.e., $r_{ij} = 0$. Depending on the value of z_{ijs}, $\mathcal{P}_3$ has the value of $\mathcal{P}_3 = \omega_{ijs} + \xi_{is}$ or $\mathcal{P}_3 = \omega_{jis} + \xi_{js}$, where $\omega_{ijs} = \omega_{jis}$. This can be generalized to include a deployment location instead of the first task. In general, $\mathcal{P}$ can be expressed by combining $\mathcal{P}_1, \mathcal{P}_2$, and $\mathcal{P}_3$:

$$\mathcal{P} = \omega_{ijs} \cdot \sum_{s\in\mathcal{S}} z_{ijs} + \mathcal{P}_2, \quad \forall i \in \mathcal{V}^{\Sigma}, j \in \mathcal{V}^{\Delta}. \tag{18}$$

Eq. (18) is a generalization of all three cases. It can easily be reduced to Eq. (16) in a case where either task i, or j, or both are virtual. By the definition of ω_{ijs} (Sect. IV), whenever at least one of the tasks is virtual $\omega_{ijs} = 0$. If task i and j can be

executed in parallel ($r_{ij} = 1$), the second part of the Eq. (18) is 0. Since tasks i and j can be run in parallel it means at least one of them is a virtual task, making the $\omega_{ijs} = 0$ and consequently $\mathcal{P} = 0$. With the combination of these three cases, we can cover a wide range of task relations.

V. A Case Study

This case study has two goals: (i) to showcase how the presented formalization of the [XD]:MT-SR-TA-SP problem configuration is mapped to a real-world scenario in practice, and (ii) to highlight the complexity, which is associated with identification of a solution, even in the simple cases.

Let us consider an underwater mission scenario with 2 agents, i.e., Autonomous Underwater Vehicles (AUVs), and 6 tasks. Out of the 6 tasks, 3 are virtual ($h_1 = h_5 = h_6 = 1$) and 3 are physical. The only tasks that can execute in parallel are task 3 and 6, i.e., $r_{36} = r_{63} = 1$.

The mission consists of measuring H_2S (hydrogen sulfide) water pollution at location A (Task 4), scanning a seabed at location B (Task 3), taking photos at location C (Task 2). The virtual tasks are as follows: Task 6 in which the gathered data from scanning the seabed (Task 3) is processed; the data processing of Task 6 can be executed in parallel with the data collection of Task 6. Also, Tasks 1 and 5, which are sending data back to the command center data gathered in Tasks 2 and 4, respectively. Furthermore, Task 5 cannot be executed before Task 4 is finished, i.e., there is a precedence constraint $4 \prec 5$. The same is applied to the Task 2 and 1, i.e., $2 \prec 1$.

Two AUVs are available for this mission. AUV 1 is equipped with a sonar, a camera system, and an acoustic modem, while AUV 2 has the equipment for water pollution measurements in addition to an acoustic modem. The final plan is shown in Fig. 1 as two sub-figures, each dedicated to an AUV. Task 0 indicates the transit from one task to another. Physical tasks are colored red, whereas virtual tasks are indicated with blue color. Transit tasks, which represent moving of an AUV from one location to the next, are colored green. As it can be seen, the plan for AUV 1 is to go to the location B and perform Task 3, while simultaneously processing the acquired data. Then AUV 1 should go to location C and perform Task 2, and send data back to the command center while going to the destination location. AUV 2 is tasked to go to location A, and take photos (Task 4). Afterwards, AUV 2 is tasked with sending photos to the command center (Task 5). This data is sent in parallel with the transit from location C to the destination location.

This small mission, which can be seen as a part of a larger mission, showcases a real-world use case of a [XD]:MT-SR-TA-SP problem configuration.

VI. Evaluation

The evaluation is performed on a set of test instances using the CPLEX solver. The experimental platform is an i9-9980XE @4.1GHz (18 cores) CPU with 128GB of DDR4 RAM. CPLEX is set with a timeout of 6 hours. We compare the presented formulation with an equivalent model that does not exploit the task parallelism, to assess the obtained advantage.

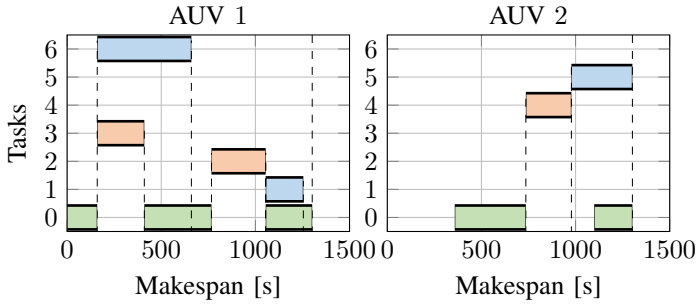

Fig. 1: The plan with parallel task execution.

A. Benchmark Settings

The benchmark consists of 6 problem instances with different levels of complexity, regarding the number of agents, tasks, and constraints. We limit the number of different equipment required by tasks to 3. The number of required precedence relations selected from the set $\{0, 1, 2\}$ through discrete uniform distribution. The number of tasks and agents vary depending on the scenario. The task duration vary from 10 to 500 seconds. Two different cases are compared on the same problem of test instances. In the first case, parallel task execution is allowed whenever it is possible. In the second case, parallel task execution is forbidden; however, virtual tasks can still be performed while agents are in transit.

B. Implementation

The performance of the CPLEX solver greatly depends on the way a model is implemented. Although the goal of this 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. In routing problems, it is common to implement sub-tour elimination constraints as lazy constraints [15]. This is the case with Eq. (8). This means that this constraint is only taken into account after CPLEX finds an incumbent solution. At this point, it is checked if this solution satisfies Eq. (8). If it does, the found solution is valid; otherwise, a new constraint is added to the model to remove this incumbent solution from the feasibility set. The disjunctive constraint Eq. (13) and the precedence constraint Eq. (12) are implemented as indicator constraints. 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 a reduced performance in terms of convergence time. Ku and Beck [16] 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.

C. Results

The obtained results from the CPLEX optimization tool for the set of problem instances, are given in Table I. Instances 1–4, take less time to compute, however as the number of agents and tasks increase, the time required for the computing the optimal solution grows exponentially, e.g., with instance 6 reaching the timeout of 6 hours, without being able to prove optimality. However, a feasible solution was always computed. Comparing the obtained solutions for the case with and without parallel execution, three things can be observed. Firstly, the computed solutions for the parallel execution cases are either better or equivalent to the cases without parallel execution. If we look at the solution of Instance 3 (Fig. 1), we have shown how the plan looks when tasks 3 and 6 can run in parallel. In Fig. 2 the same mission is presented, but without allowing the parallel execution of tasks. The overall makespan is shorter when parallelism is allowed. In Instances 2, 3, 4, and 6, the makespan is decreased up to $\sim$ 20%, when the parallelism is exploited. It is to be noticed that such numeric advantages can be more or less based on the specific problem setup (number of agents, number of tasks, precedence constraints, parallelism, etc.), and hence is not trivial to predict *a priori*. For Instances 1 and 5, the computed solutions are the same in both cases with and without parallel task execution. In general, this can happen whenever the virtual tasks can be completed during the transits, and therefore the serialized execution has the same makespan as its parallel counterpart.

VII. Conclusion

We have presented a mathematical model for the optimization of multi-robot mission planning, which is able to exploit task parallelism. In particular, we have focused on the formulation of the [XD]:MT-SR-TA-SP problem configuration as a MILP problem, with the objective of minimizing the total makespan of the mission. The presented model has been evaluated over a set of problem instances using CPLEX, in order to reveal both the generality of the proposed approach, and the advantages associated with the exploitation of task parallelism. Furthermore, we have introduced the distinction of virtual and physical tasks, characterizing their relations in terms of parallel task execution.

Future work will extend this formulation to include Time-Windows and Multi-Robot tasks. Additionally, further investigation is needed to provide more scalable solvers for the presented problems, possibly trading off optimality for feasibility. Thus, meta-heuristic approaches, as well as constraint-programming, pose a sensible option for a future solver design.

TABLE I: Results for parallel and non-parallel task execution.

	Parallel execution			Non-parallel execution		
Inst.	**Optimal**	**Cost**	**Time [s]**	**Optimal**	**Cost**	**Time [s]**
1	Yes	1056	10.9	Yes	1056	13.1
2	Yes	1128	14.5	Yes	1312	21.2
3	Yes	1302	0.3	Yes	1445	0.3
4	Yes	752	16.9	Yes	901	282.4
5	Yes	735	4623	Yes	735	6323
6	No	1093	21613	No	1330	21619

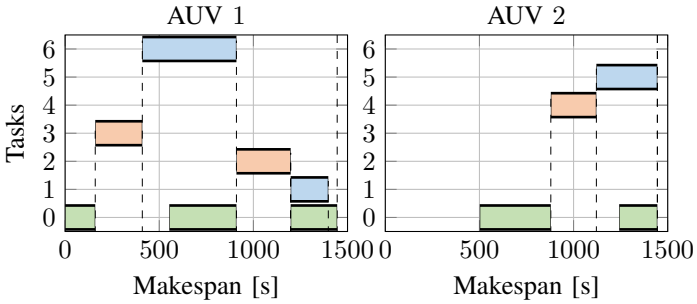

Fig. 2: The plan without parallel task execution.

References

[1] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *Ieee Access*, vol. 6, pp. 28 573–28 593, 2018.

[2] B. Miloradović, B. Çürüklü, M. Ekström, and A. V. Papadopoulos, "A genetic algorithm approach to multi-agent mission planning problems," in *Operations Research and Enterprise Systems*, 2020, pp. 109–134.

[3] 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 *AAMAS*, 2020, pp. 375–383.

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

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

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

[7] F. C. Van Delft, G. Ipolitti, D. V. Nicolau Jr, 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, p. 20180034, 2018.

[8] P. Toth and D. Vigo, "The vehicle routing problem," *Discrete Mathematics and Applications, Society for Industrial and Applied Mathematics, Philadelphia*, 2002.

[9] K. Jansen, M. Mastrolilli, and R. Solis-Oba, "Approximation schemes for job shop scheduling problems with controllable processing times," *European journal of OR*, vol. 167, no. 2, pp. 297–319, 2005.

[10] 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 *ICRA*, 2014, pp. 5414–5419.

[11] 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 *ICAPS*, vol. 29, 2019, pp. 707–715.

[12] B. Miloradović, M. Frasheri, B. Cürüklü, M. Ekström, and A. V. Papadopoulos, "Tamer: Task allocation in multi-robot systems through an entity-relationship model," in *PRIMA*. Springer, 2019.

[13] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large-scale traveling-salesman problem," *Journal of the Operations Research Society of America*, vol. 2, no. 4, pp. 393–410, 1954.

[14] M. Gini, "Multi-robot allocation of tasks with temporal and ordering constraints," in *AAAI*, 2017.

[15] M. M. Aguayo, S. C. Sarin, and H. D. Sherali, "Solving the single and multiple asymmetric traveling salesmen problems by generating subtour elimination constraints from integer solutions," *IISE Transactions*, vol. 50, no. 1, pp. 45–53, 2018.

[16] W.-Y. Ku and J. C. Beck, "Mixed integer programming models for job shop scheduling: A computational analysis," *Computers & Operations Research*, vol. 73, pp. 165–173, 2016.