Human-Multi-Robot Task Allocation in Agricultural Settings: a Mixed Integer Linear Programming Approach

Martina Lippi, Jorand Gallou, Jozsef Palmieri, Andrea Gasparri, Alessandro Marino

***Abstract*— The use of heterogeneous human-multi-robot teams enables the combination of complementary skills of these two different types of agents. To have an effective collaboration, it is necessary to define a strategy for allocating and scheduling tasks among them. In this work, we distinguish robots in working robots and service ones: working robots and human operators can perform similar tasks in the environment and both are assisted by service robots. We propose a Mixed-Integer Linear Programming approach that aims to minimize the waiting times of the working agents, the energy consumption of the service robots, and the makespan while ensuring that the velocity constraints of the robots are met and the task ordering is correct. Furthermore, we propose an online updating strategy that tackles changes in the parameters of working agents and adapts the plan accordingly based on a heuristic algorithm. To validate our framework, we analyze a precision agriculture harvesting application with two human operators, two working robots, and two service robots.**

I. Introduction

In recent years, robots have become increasingly prevalent in a wide range of fields thanks to their growing capacity for context awareness and reasoning. However, collaboration with human operators remains pivotal in many areas to achieve successful outcomes. This is particularly true in dynamic and unstructured environments where full autonomy of the robots is challenging to realize [1]. Examples of such environments include precision agriculture, search and rescue operations, and healthcare settings, to name a few. Within the realm of human-robot scenarios, we narrow our focus to human-multi-robot settings where robots are capable of carrying out tasks autonomously or assisting human operators. In this context, a defined number of tasks must be completed, and our aim is to explore the most effective ways to distribute and schedule these tasks among the available robots and human operators [2]. In doing this, it is necessary to take into account the variable human behavior, which might be influenced by several factors such as fatigue, distraction, and changing task requirements [3].

Mixed Integer Linear Programming (MILP) formulations represent a valuable tool for finding an optimal assignment of tasks while taking into account possible (linear) constraints of the problem at hand [4]. Several approaches exist in the literature which have explored the use of MILP for minimizing the overall execution time, i.e., the makespan, in collaborative settings. An example can be found in [5] where multiple robots and a human operator are considered in the MILP formulation.

M. Lippi, J. Gallou and A. Gasparri are with Roma Tre University, Italy, {*martina.lippi,jorand.gallou*}*@uniroma3.it, gasparri@dia.uniroma3.it*.

A. Marino is with University of Cassino and Southern Lazio, Italy, {*al.marino, jozsef.palmieri*}*@unicas.it*.

This work has been supported by the European Commission under the grant agreement number 101016906 – Project CANOPIES.

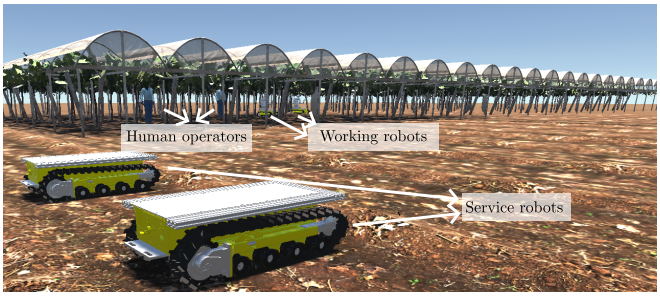

Fig. 1: Illustration of the considered scenario.

A discretization of the time horizon is used which makes it inefficient for long-horizon planning or fine discretization settings. A set of discrete time points is also exploited in [6] where a single-robot single-human scenario is analyzed. In addition to optimizing the makespan, ergonomics aspects are considered as well to minimize human fatigue. The time-horizon discretization is overcome, for instance, in the formulation proposed in [7], where additional cost indices based on switching costs and quality measures are defined. Several features such as the capability for humans to supervise robot operations when autonomy is not possible are also included. A MILP formulation is additionally proposed in [8] where the human workload is taken into account. In this formulation, a prior subdivision of the tasks into several layers is made based on precedence constraints and the cycle time of these layers is minimized. A similar layer architecture of the tasks is exploited in [9] where priorities are set on the basis of the task dependency order.

Further methodologies for task allocation and scheduling in human-robot collaborative settings can be found in the literature. For instance, an AND/OR graph representing the collaborative job is employed in [10] to design a scheduler that is online locally updated in case of failures. A chemical reaction optimization algorithm is designed in [11] to include micro-breaks within job-cycles, allowing recovery from fatigue. Interestingly, a discrete Bee Algorithm is adopted in [12] to realize a cooperative disassembly task by minimizing human fatigue. A genetic algorithm is proposed in [13] for an assembly task. Specifically, a simulation tool is realized and exploited to optimize a fitness function based on task progress, waiting time, and traveled distance. Multi-agent deep reinforcement learning has also been applied for a collaborative assembly task in [14]. However, many of the above approaches do not take into account the variability in the human parameters.

In this work, we consider a heterogeneous human-multi-robot setting involving: (*i*) human operators and working robots, that can perform operations in the environment, and (*ii*) service robots, that provide assistance to them. Taking in-

spiration from the H2020 European project CANOPIES, we examine a precision agriculture scenario during the table grape harvesting season as depicted in Figure 1. Specifically, we consider working robots and human operators that carry out box filling operations with table grape bunches, assisted by service robots performing box exchange and transporting services. The proposed approach involves formulating a MILP-based optimization problem that defines the allocation of service tasks to the robots and their scheduling while minimizing a combination of waiting time, makespan, and robot energy. Notably, our approach differs from many previous works in that we also consider the velocity of the service robots as an output of the optimization problem. Moreover, we take into account possible changes in the parameters of the working agents, including human operators, and design an online updating strategy that reacts to such changes. We build on our previous work [15], extending it to a more complex scenario and a more general formulation with and including a novel replanning strategy. We validate the approach in a realistic simulator with two humans, two working robots, and two service robots.

II. Preliminaries

The considered scenario involves a heterogeneous human-multi-robot team where we identify *i)* human operators and *ii)* working robots (which can be equipped, for instance, with manipulators), that carry out multiple consecutive operations within the environment, and *iii)* mobile service robots, that provide assistance to the operations. We refer to the first two categories as working agents. We assume that every time a working agent completes an operation, a service is needed to enable the start of the subsequent operation. Furthermore, we assume that a depot station is present where the service robots start from and return to after every service. For example, in an agricultural setting, human operators and working robots may perform the task of harvesting fruits and filling boxes with multiple fruits. Meanwhile, service robots may aid them by replacing the full fruit boxes with empty ones and transporting the full boxes to a designated depositing station. A depiction of the considered scenario is provided in Figure 1. In this setup, we are interested in defining a strategy for allocating and scheduling the service activities in such a way to ensure a correct execution of the tasks and optimize the performance of the overall system, while taking into account possible variations in the agents' parameters.

In the following, a formal definition of the overall system is presented. Given a set $\mathcal{S} = \{s_1, s_2, .., s_n\}$, we adopt the following notations: $i \in \mathcal{S}$ denotes the i-th element of the set, and $|\mathcal{S}|$ represents the cardinality of the set. If the set includes elements with two indices, such as $\mathcal{S} = \{s_{1,2}, s_{1,2}, .., s_{n,m}\}$, we use the notation $(i, j) \in \mathcal{S}$ to refer to the element $s_{i,j}$. As described above, we have a set of mobile service robots, denoted by $\mathcal{M}$, and a set of working agents to assist, denoted by $\mathcal{A}$. The latter set includes working robots, composing the set $\mathcal{W}$, and human operators, composing the set $\mathcal{H}$, i.e., it holds $\mathcal{A} = \mathcal{H} \cup \mathcal{F}$. We denote by n^m, n^a, n^h, and n^w the number of service robots, working agents, human operators and working robots, respectively, i.e., it holds $|\mathcal{M}| = n^m$, $|\mathcal{A}| = n^w$, with $n^a = n^h + n^w$, $|\mathcal{H}| = n^h$, and $|\mathcal{W}| = n^w$. We model the environment as a two-dimensional space, where the subspace external to the depot is referred to as working area. Each service robot $m \in \mathcal{M}$ is assigned a specific location at the depot, which we refer to as the base position and indicate with b_m. Similarly, for each working agent $a \in \mathcal{A}$, the respective position is represented by x_a. The path length required to reach the agent a from the base position of the service robot m is denoted by $l_{m,a}$, which is equivalent to the length of the return journey from the agent to the robot's base position. We denote with $l_{\max} = \max_{m \in \mathcal{M},\, a \in \mathcal{A}} l_{m,a}$ the maximum possible path length to traverse. Note that, in general, the position of the working agents within the environment can vary over time, implying that the length variables can also be time-varying.

Additionally, for each mobile service robot m, we take into account the minimum and maximum cruising velocities, which are denoted by $v_{\min,m}$, for which it holds $v_{\min,m} > 0$ and $v_{\max,m}$, for which it holds $v_{\max,m} \geq v_{\min,m}$, respectively. The maximum velocity is determined by the physical limitations of the robot, while the minimum velocity is influenced by both physical constraints (e.g., to overcome static friction) and the need to limit the robot's presence in the working area. We denote with $v_{\min} = \min_{m \in \mathcal{M}} v_{\min,m}$ and $v_{\max} = \max_{m \in \mathcal{M}} v_{\max,m}$ the minimum and maximum allowed velocity among all robots, respectively.

As far as the working agents are concerned, every agent a carries out a set of q_a ordered operations. Let $\tau^o_{a,i}$ be the i th operation performed by the agent a, we denote its start time and end time as $\underline{o}_{a,i}$ and $\overline{o}_{a,i}$, respectively. We indicate the ordered set of operations of the same agent with $\mathcal{O}_a = \{\tau^o_{a,1}, ..., \tau^o_{a,q_a}\}$, where it holds that $\underline{o}_{a,i} \geq \overline{o}_{a,i-1}$, $\forall i \in \{2, ..., q_a\}$. In our case study, as an example, an individual operation corresponds to filling a box with fruits in an agricultural scenario. By considering all human and robotic working agents, we obtain the collective set of operations as $\mathcal{O} = \mathcal{O}_1 \cup ... \cup \mathcal{O}_m$. To simplify our notation, we assume that all q_a operations executed by an agent a have the same duration δ^o_a. However, our framework easily accommodates scenarios where the duration of each operation may differ. Similar to the working agent positions, it is important to note that also the duration variables may vary over time. For instance, human operators might be faster or slower at working depending on the level of fatigue and/or stress, while working robots may have different operating speeds depending on the variability of the environment in which they are deployed. This variability can make it more or less complex for the robots to carry out operations effectively, thus making them potentially slower or faster.

For each operation $\tau_{a,i} \in \mathcal{O}$, we define a respective assistance activity that a service robot has to carry out. This is composed of four consecutive phases, also referred to as tasks: *i)* a *going* phase $\tau^g_{a,i}$, where the robot, starting from its depot, navigates the field to go in the proximity of the working agent a; *ii)* a *waiting* phase $\tau^w_{a,i}$, where the robot waits for the completion of i th operation of the agent a to carry out the service; *iii)* a *serving* phase $\tau^s_{a,i}$, where the robot actually performs the service for the agent; *iv)* a *depositing* phase $\tau^d_{a,i}$, where the robot returns to the base position at the depot and performs

any conclusive action if needed (e.g., releasing a load at the depot). To indicate the robot performing the assistance activity (i.e., the four phases above), we introduce the binary decision variable $S_{a,i,m} \in \{0,1\}$, $\forall (a,i) \in \mathcal{O}, m \in \mathcal{M}$, that is equal to 1 when the mobile service robot m serves the i th operation of the working agent a, and is 0 otherwise. Clearly, when $S_{a,i,m} = 1$, it means that the mobile robot m has to carry out all four phases of the respective assistance activity. Additionally, we denote the start times of the above four tasks as $\underline{g}_{a,i}$, $\underline{w}^s_{a,i}$, $\underline{s}_{a,i}$, and $\underline{d}_{a,i}$, respectively, and the corresponding end times as $\overline{g}_{a,i}$, $\overline{w}^s_{a,i}$, $\overline{s}_{a,i}$, and $\overline{d}_{a,i}$, respectively. We use δ^s_m to represent the time needed for the robot m to perform a service at a working agent position, and δ^d_m to represent the time to perform a conclusive action at the depot. For instance, in our case study, the service task involves box replacement, i.e., the robot is loaded with a filled box and provides an empty one to the working agent, while, during the depositing phase, the conclusive action involves a box emptying, i.e., the robot releases the full box at the depot. We denote with $\delta^s_{\max} = \max_{m \in \mathcal{R}} \delta^s_m$ and $\delta^d_{\max} = \max_{m \in \mathcal{R}} \delta^d_m$ the maximum service time and final action at the deposit among all robots, respectively.

III. Problem statement and solution overview

To determine how to allocate and schedule the service activities, we introduce relevant cost indices. First, we consider the waiting time of each working agent, representing the agent idle time during which no operation can be carried out. Second, we take into account the energy consumption of the service robots for navigating the environment. Third, we consider the overall makespan, representing the overall time to complete all tasks.

As far as the waiting time of the working agents is concerned, let $w^o_{a,i}$ be the normalized waiting time associated to the operation $\tau_{a,i} \in \mathcal{O}$. We compute it as follows

$$w^o_{a,i} = \frac{\left(\underline{o}_{a,i} - \overline{s}_{a,i-1}\right)}{l_{\max}/v_{\min} + \delta^s_{\max}} \quad \text{with } i > 1, \tag{1}$$

i.e., it is given by the difference between the starting time of the operation and the end time of the previous service task by a robot, normalized with respect to the maximum time that a robot may need to reach the working agent and perform the service. For instance, in our case study, in order for a working agent to begin filling a new box, a service robot must replace the previously filled box with an empty one. For the first operation of each working agent, we consider zero waiting time, i.e., $w^o_{a,1} = 0, \forall a \in \mathcal{A}$.

As far as the energy contributions are concerned, we first introduce the average traveling velocities of the service robots. Specifically, we denote by $v^g_{a,i}$ the average velocity of the robot reaching the agent a for the i th operation, and we obtain it as

$$v^g_{a,i} = \textstyle\sum_{m \in \mathcal{M}} S_{a,i,m} l_{m,a} / (\overline{g}_{a,i} - \underline{g}_{a,i}), \tag{2}$$

that is the ratio between the distance traveled by the serving robot (i.e., the one with $S_{a,i,m} = 1$) to reach the agent and the time taken to complete the travel. Similarly, we define the average velocity to return to the base position $v^d_{a,i}$ as

$$v^d_{a,i} = \frac{\sum_{m \in \mathcal{M}} S_{a,i,m} l_{m,a}}{\overline{d}_{a,i} - \sum_{m \in \mathcal{M}} S_{a,i,m} \delta^d_m - \underline{d}_{a,i}}, \tag{3}$$

where the term $\sum_{m \in \mathcal{M}} S_{a,i,m} \delta^d_m$ represents the time spent at the depot station and is not included in the robot's travel time. Based on these quantities, we define an energy-like term $e_{a,i}$ which increases with higher traveling velocities [16] as follows

$$e_{a,i} = \left(e^g_{a,i} + e^d_{a,i}\right)/2, \tag{4}$$

where

$$e^g_{a,i} = k_e \left(\frac{v^g_{a,i} - v_{\min}}{v_{\max} - v_{\min}}\right), \; e^d_{a,i} = k_e \left(\frac{v^d_{a,i} - v_{\min}}{v_{\max} - v_{\min}}\right),$$

with k_e a positive constant.

As far as the makespan is concerned, we indicate it with the variable Δ and obtain its normalized version as

$$\Delta = \frac{\max_{(a,i) \in \mathcal{O}} \overline{d}_{a,i} - t_0}{n^a (2 l_{\max}/v_{\min} + \delta^s_{\max} + \delta^d_{\max})}, \tag{5}$$

obtained by considering that all tasks are executed at the slowest speed and where t_0 is the initial time of the experiment.

We can now state the main problem addressed in this work.

Problem 1. Consider a system composed of n^a working agents in the set $\mathcal{A}$, including n^h human operators and n^w working robots, with positions x_a, $\forall a \in \mathcal{A}$. The working agents realize operations $\mathcal{O}$ in the environment with agent-dependent duration δ^o_a. Consider n^m mobile service robots in the set $\mathcal{M}$ with base positions b_m at the depot, $\forall m \in \mathcal{M}$, assisting the working agents. Each assistance activity comprises four tasks: going to the working agent, waiting for him/her/it, serving, and returning to the depot. Our goal is to plan the allocation and scheduling of all the tasks, i.e., the binary decision variables $S_{a,i,m}, \forall (a,i) \in \mathcal{O}, m \in \mathcal{M}$ and the real decision variables $\underline{g}_{a,i}$, $\underline{w}^s_{a,i}$, $\underline{s}_{a,i}$, $\underline{d}_{a,i}$, $\underline{g}_{a,i}$, $\underline{o}_{a,i}$, $\overline{w}^s_{a,i}$, $\overline{s}_{a,i}$, $\overline{d}_{a,i}$, and $\overline{o}_{a,i}$, while minimizing a combination of working agent waiting time, robot energy and makespan. The plan must be able to adapt to variations over time of the positions and operation durations of the working agents, which can occur particularly for humans.

We address the above problem with a two-layer architecture. First, an optimal allocation and scheduling layer is responsible for generating the plan; second, an updating layer is in charge of modifying the plan according to variations of the working agent parameters. Regarding the optimal solution, we formulate a MILP problem generating the allocation and scheduling variables by assuming that all the parameters of the working agents are constant. Then, we take into account possible variations in these parameters in the updating strategy. Specifically, we assume that a monitoring module is available that estimates position and duration of the working agents. Based on these values, we establish whether the same allocation can be preserved, by only possibly re-scheduling the tasks, or if a new optimal solution should be computed. Note that the development of a monitoring module is beyond the scope of the paper, but many approaches exist in the literature that can be adopted for this purpose, such as [17]. Finally, note that by construction a feasible solution to Problem 1 can always be found.

IV. MILP Formulation

In this section, we focus on the formulation of the MILP problem. The inputs of the problem consist of, $\forall m \in \mathcal{M}, a \in \mathcal{A}$: the minimum and maximum cruising velocities of the service robots, $v_{\min,m}, v_{\max,m}$, the path lengths

$l_{m,a}$, the times δ^s_m and δ^d_m for the service and depositing actions, the number q_a of operations of each working agent and the duration δ^o_a. Based on these inputs, the allocation and scheduling decision variables must be determined. According to Problem 1, we aim to minimize the following cost function

$$c = \alpha \sum_{\substack{(a,i)\in\mathcal{O},\\ a\in\mathcal{H}}} w^o_{a,i} + \beta \sum_{\substack{(a,i)\in\mathcal{O},\\ a\in\mathcal{W}}} w^o_{a,i} + \gamma \sum_{(a,i)\in\mathcal{O}} e_{a,i} + \kappa\Delta, \quad (6)$$

with $\alpha, \beta, \gamma, \kappa$ positive weights. The cost function is modeled in such a way to achieve a balance between the different relevant cost indices. Specifically, it aims to minimize the waiting time of both humans and working robots, in order to reduce their idle time. Additionally, it seeks to minimize the energy consumption of service robots, extend their lifespan and reduce costs. Finally, it aims to minimize the overall makespan, which is particularly relevant for the depositing tasks, for which there is no subsequent waiting time of a working agent. It is evident that some of these indices are in opposition to each other. For instance, to decrease the waiting time for humans, it might be necessary to increase the velocity of the service robots, which results in greater energy consumption. Hence, the balance that is attained by the solution depends on the selection of weights.

The following constraints are identified.

1) Assignment of the assistance tasks :

$$\textstyle\sum_{m\in\mathcal{M}} S_{a,i,m} = 1, \quad \forall (a,i) \in \mathcal{O}. \quad (7)$$

This guarantees that each operation (a,i) of a working agent is served by exactly one service robot, i.e., among all robots in $\mathcal{M}$, only one of them should satisfy the condition $S_{h,i,m} = 1$.

2) Duration for going tasks:

$$\overline{g}_{a,i} - \underline{g}_{a,i} \geq S_{a,i,m} l_{m,a}/v_{\max,m}, \quad (8a)$$

$$\overline{g}_{a,i} - \underline{g}_{a,i} \leq S_{a,i,m} l_{m,a}/v_{\min,m}, \quad (8b)$$

$\forall (a,i) \in \mathcal{O}, m \in \mathcal{M}$. The above constraints guarantee that the time duration scheduled for each going task does not violate the limitations on the traveling velocity of the robots. Specifically, the first inequality ensures that, for each going task, the scheduled duration is above the minimum travel time required by the robot m assigned to the task (having $S_{a,i,m} = 1$). This minimum travel time is obtained as $l_{m,a}/v_{\max,m}$, i.e., by considering the maximum speed of the robot during the traveling. Similarly, the second inequality guarantees that the duration is below the maximum travel time $l_{m,a}/v_{\min,m}$.

3) Duration for depositing tasks :

$$\overline{d}_{a,i} - \underline{d}_{a,i} \geq S_{a,i,m} l_{m,a}/v_{\max,m} + S_{a,i,m}\delta^d_m, \quad (9a)$$

$$\overline{d}_{a,i} - \underline{d}_{a,i} \leq S_{a,i,m} l_{m,a}/v_{\min,m} + S_{a,i,m}\delta^d_m, \quad (9b)$$

$\forall (a,i) \in \mathcal{O}, m \in \mathcal{M}$. Similar to the previous constraints, the above inequalities guarantee that the limits on the robot velocities are satisfied for each depositing task. In this case, the duration of the task also includes the time for the possible final depositing action δ^d_m. In our case study, this time corresponds to the time for emptying the full box at the depot station.

4) Sequence of the assistance tasks:

$$\begin{aligned} \underline{w}^s_{a,i} &= \overline{g}_{a,i}, \\ \underline{s}_{a,i} &= \overline{w}^s_{a,i}, \quad \forall (a,i) \in \mathcal{O} \\ \underline{d}_{a,i} &= \overline{s}_{a,i}, \end{aligned} \quad (10)$$

The above equalities establish the correct sequence of the assistance phases. Specifically, the first constraint states that, for each operation $(a,i) \in \mathcal{O}$, the start time of the waiting phase (i.e., $\underline{w}^s_{a,i}$) coincides with the end of the going phase (i.e., $\overline{g}_{a,i}$), meaning that the waiting phase is immediately consecutive to the going one. Similarly, the second and third constraints ensure that the service phase follows the waiting one, and the depositing phase follows the service one, respectively.

5) End time of service tasks:

$$\overline{s}_{a,i} \geq \overline{o}_{a,i} + \textstyle\sum_{m\in\mathcal{M}} S_{a,i,m}\delta^s_m, \quad \forall (a,i) \in \mathcal{O}. \quad (11)$$

This inequality allows to properly set the end time of the service tasks. In particular, it requires, for each operation $(a,i) \in \mathcal{O}$, that the end time of the service task is greater than or equal to the end time of the operation performed by the working agent a plus the service time of the robot assigned to the task (i.e., having $S_{a,i,m} = 1$). This guarantees that, to serve a working agent, the latter must have completed the operation of interest.

6) Sequence of operations of working agents:

$$\underline{o}_{a,1} = t_0, \qquad \forall a \in \mathcal{A} \quad (12a)$$

$$\overline{o}_{a,i} = \underline{o}_{a,i} + \delta^o_a, \qquad \forall (a,i) \in \mathcal{O} \quad (12b)$$

$$\underline{o}_{a,i} \geq \overline{s}_{a,i-1}, \qquad \forall (a,i) \in \mathcal{O} \quad (12c)$$

These constraints determine the sequence of the operations performed by the working agents. Specifically, the first equation sets the start time of the first operation of each agent to the initial time of the experiment. The second equality imposes that the end time of each operation (a,i) (i.e., $\overline{o}_{a,i}$) is given by the corresponding start time (i.e., $\underline{o}_{a,i}$) plus the respective duration δ^o_a. Finally, the last inequality ensures that a working operator cannot start an operation if the previous one has not been served. This is encoded by enforcing that the start time of the operation (a,i) must be greater than or equal to the end time of the service task $(a,i-1)$.

7) Execution of one task at a time:

$$\begin{aligned} \underline{z}_{p,k} - \overline{u}_{a,i} \geq &-K(2 - S_{a,i,m} - S_{p,k,m}) \\ &-K(1 - Q^{uz}_{a,i,p,k,m}), \end{aligned} \quad (13a)$$

$$\underline{u}_{a,i} - \overline{z}_{p,k} \geq -K(2 - S_{a,i,m} - S_{p,k,m}) - K\, Q^{uz}_{a,i,p,k,m}, \quad (13b)$$

$\forall m \in \mathcal{M}, (a,i), (s,k) \in \mathcal{O}$ with $(a,i) \neq (s,k)$, where K is an arbitrarily large constant, and, with an abuse of notation, we consider $u, z \in \{g, w^s, s, d\}$ meaning that, for instance, when $z = g$ the variable $\underline{z}_{p,k}$ corresponds to $\underline{g}_{p,k}$. In addition, $Q^{uz}_{a,i,s,k,m} \in \{0,1\}$ is an auxiliary binary decision variable detailed in the following. The above constraints ensure that each service robot only executes a task at a time, i.e., it cannot assist multiple working agents at the same time. Specifically, let us analyze the case when a service robot m has to provide assistance for two operations (a,i) and (p,k) (i.e., $S_{a,i,m} = S_{p,k,m} = 1$), and let us consider, for instance, $z = g$ and $u = d$. In this case, the above constraints enforce that either it holds that the going task for (p,k) is executed after the depositing task (a,i) is completed, i.e., $\underline{g}_{p,k} \geq \overline{g}_{a,i}$ with $U^{gd}_{a,i,p,k,m} = 1$, or the opposite, i.e., $\underline{g}_{a,i} \geq \overline{g}_{p,k}$ with $U^{gd}_{a,i,p,k,m} = 0$. In the case the robot m is not assigned to both the operations (i.e., either it holds $S_{a,i,m} = 0$ or $S_{p,k,m} = 0$), no constraints are

enforced by (13a) and (13b) for the relative start and end times.

Note that the proposed formulation is not limited to precision agriculture settings but is applicable to different service tasks and settings, such as assembly tasks in industrial contexts.

V. Online updating strategy

When solving the MILP problem, we assume that all the parameters of the working agents remain constant over time. However, especially regarding the human operators, it is evident that changes in their parameters, i.e., duration variables δ_h^a and locations x_a, can occur. To address this issue, we design an online updating strategy. Briefly, once a variation is detected, an update algorithm is designed to adjust the times of the tasks and operations accordingly in order not to violate any constraint. For instance, if a person starts working at a slower speed than at the beginning, this may lead to invalid times for the corresponding robot services. Afterward, we evaluate the cost function of the updated plan and compare it with the one obtained from the original schedule. If the difference in cost exceeds a specified threshold, we recalculate the solution by resolving the optimization problem. It is important to note that when a change is detected, we do not want to automatically simply recalculate the optimal solution every time, since this may lead to constantly apply major changes in the plan. This may be not desirable especially in tasks where human operators are involved, where these updates may cause interruptions and disrupt their workflow, leading to frustration and decreased productivity [18]. Furthermore, since MILP problem are NP-hard, continuously re-compute the optimal solution might not be feasible. This aspect could be mitigated by implementing appropriate heuristics as, for example, in [19], which however is out of the scope of this paper. In the following, we denote with t_{opt} the time at which the most recent optimal solution was obtained by solving the MILP problem. Then, we denote with t_{use} the time when the currently used plan was last computed or updated. Clearly, at the beginning $t_{use} = t_{opt}$. Finally, we use t_{ch} to represent the time at which a change in the parameters was detected. Only the tasks that have not been completed by time t_{ch} are considered in the following. The notation $y(t)$ is used to represent the value of variable y at a specific time t.

A. Update of the plan

Algorithm 1 summarizes the main steps for updating the plan given the agent a^* with changing parameters. At the beginning, we initialize the plan at time t_{ch} with the last one computed at time t_{use} (line 1). For the simplicity of notation when we omit the time variable, we refer to t_{ch}. We start analyzing the first operation of the agent a^* that has not been completed yet at time t_{ch} and loop over all the subsequent ones. We update the final time of the current operation (a^*, i) according to the duration $\delta^o_{a^*}(t_{ch})$ (line 3) and get the index m^* of the service robot assisting the operation (i.e., having $S_{a^*,i,m^*} = 1$) in line 4. In addition, we initialize to false a boolean variable that will be used to determine whether a checking must be done on shifting the scheduled times. At this point, we differentiate the analysis on whether a change in the operation duration or in the lengths has been recorded. In the first case, we check if the start time of the respective service task is lower than the updated end time of the operation (line 6), i.e., the service robot wants to serve the operation (a^*, i) before it has been completed. If this condition is met, we need to update the plan. Specifically, we first set the end of the waiting task and the start time of the service task related to (a^*, i) equal to $\overline{o}_{a^*,i}$ and then update the end time of the service task and the start time of the depositing one accordingly (line 7). We additionally set the boolean variable to true (line 8). In the case of variation in the working agent position, we check if the path length has been increased (thus potentially leading to constraint violations) and if the service has not been completed by time t_{ch}. If these conditions are met, we first try to speed up the going phase of the robot performing the assistance activity by considering the maximum velocity $v_{\max,m^*}$ (line 10). If speeding up is not possible or not sufficient, we update the final time of the going task, as well as the respective waiting and service tasks and the start time of the depositing task (lines 11-12). Similar to the previous case, we set the boolean variable to true. From line 14, we start checking if shifting the subsequent tasks is necessary in case the boolean variable is equal to true. More in detail, we first update the starting time of the next operation of the working robot a^* if necessary (lines 15-16). Then, we check if the robot can speed up during the depositing phase (leaving $\overline{d}_{a^*,i}$ unchanged) in order not to update the subsequent tasks

Algorithm 1 Updating Algorithm

Input: agent a^* with updated parameters

1: plan(t_{ch}) $\leftarrow$ plan(t_{use})
2: **for** $i \in \{1, ..., q_{a^*}\}$ **and** $\overline{o}_{a^*,i} < t_{ch}$ **do**
3: $\quad \overline{o}_{a^*,i} \leftarrow \underline{o}_{a^*,i} + \delta^o_{a^*}(t_{ch})$
4: $\quad m^* \leftarrow$ assigned robot(S, a^*, i)
5: $\quad$ check shift $\leftarrow false$
6: $\quad$ **if** $\delta^o_{a^*}(t_{ch}) \neq \delta^o_{a^*}(t_{use})$ **and** $\underline{s}_{a^*,i} < \overline{o}_{a^*,i}$ **then**
7: $\quad\quad \overline{w}^s_{a^*,i} \leftarrow \overline{o}_{a^*,i}, \quad \underline{s}_{a^*,i} \leftarrow \overline{o}_{a^*,i}, \quad \overline{s}_{a^*,i} \leftarrow \underline{s}_{a^*,i} + \delta^s_{m^*}, \quad \underline{d}_{a^*,i} \leftarrow \overline{s}_{a^*,i}$
8: $\quad\quad$ check shift $\leftarrow true$
9: $\quad$ **else if** $l_{m^*,a^*}(t_{ch}) > l_{m^*,a^*}(t_{use})$ and $\overline{s}_{a^*,i} > t_{ch}$ **then**
10: $\quad\quad$ **if** $\overline{g}_{a^*,i} - \underline{g}_{a^*,i} \leq l_{m^*,a^*}/v_{max,m^*}$ {*Going task speeding up not possible*} **then**
11: $\quad\quad\quad \overline{g}_{a^*,i} \leftarrow \underline{g}_{a^*,i} + l_{m^*,a^*}/v_{max,m^*}$
12: $\quad\quad\quad \underline{w}^s_{a^*,i}, \overline{w}^s_{a^*,i}, \underline{s}_{a^*,i} \leftarrow \overline{g}_{a^*,i}, \quad \overline{s}_{a^*,i} \leftarrow \underline{s}_{a^*,i} + \delta^s_{m^*}, \quad \underline{d}_{a^*,i} \leftarrow \overline{s}_{a^*,i}$
13: $\quad\quad\quad$ check shift $\leftarrow true$
14: $\quad$ **if** check shift **then**
15: $\quad\quad$ **if** $i < q_{a^*}$ and $\underline{o}_{a^*,i+1} < \overline{s}_{a^*,i}$ **then**
16: $\quad\quad\quad \underline{o}_{a^*,i+1} \leftarrow \overline{s}_{a^*,i}$
17: $\quad\quad$ **if** $\overline{d}_{a^*,i} - \underline{d}_{a^*,i} \leq l_{m^*,a^*}/v_{max,m^*} + \delta^d_{m^*}$ {*Depositing task speeding up not possible*} **then**
18: $\quad\quad\quad \overline{d}^{prev}_{a^*,i} \leftarrow \overline{d}_{a^*,i}$,
19: $\quad\quad\quad \overline{d}_{a^*,i} \leftarrow \underline{d}_{a^*,i} + l_{m^*,a^*}/v_{max,m^*} + \delta^d_{m^*}$
20: $\quad\quad\quad \eta \leftarrow \overline{d}_{a^*,i} - \overline{d}^{prev}_{a^*,i}$ {*Shifting quantity*}
21: $\quad\quad\quad$ **for** $r \in \mathcal{M}$ **do**
22: $\quad\quad\quad\quad \mathcal{T}_r \leftarrow$ ordered assistance operations robot (r, S)
23: $\quad\quad\quad\quad$ **for** $(a_r, i_r) \in \mathcal{T}_r$ consecutive to (a^*, i) **do**
24: $\quad\quad\quad\quad\quad$ plan $\leftarrow$ shift assistance tasks (a_r, i_r, η)
25: **for** $(a, i) \in \mathcal{O}$, with $i > 1$ **do**
26: $\quad$ **if** $\underline{o}_{a,i} < \overline{s}_{a,i-1}$ **then**
27: $\quad\quad \underline{o}_{a,i} \leftarrow \overline{s}_{a,i-1}, \quad \overline{o}_{a,i} \leftarrow \underline{o}_{a,i} + \delta^o_a$
28: $t_{use} \leftarrow t_{ch}$

(line 15). If this is not possible, we store the current depositing end time in a variable $\overline{d}^{prev}_{a^*,i}$ (line 16) and update it by assuming the maximum velocity of the service robot (line 17). Next, we compute the shifting quantity $\eta = \overline{d}_{a^*,i} - \overline{d}^{prev}_{a^*,i}$ (line 17) and update the depositing end time (line 18). At this point, we shift all consecutive assistance tasks: for each service robot r, we get the set $\mathcal{T}_r$ of ordered operations that the robot assists (line 22) and, for each operation (a_r, i_r) consecutive to (a^*, i), we apply a shift of η for the corresponding assistance tasks.

After analyzing all operations of the agent a^*, we finally update the start and end times of the working agents if needed (lines 25-27) and set the variable t_{use} equal to t_{ch}.

B. Strategy for re-computation

To evaluate if a re-computation of the MILP solution is needed, we evaluate the cost variation compared to the one generated at time t_{opt}. As stated above, we only focus on the tasks which are not completed by time t_{ch} and denote with C^+ the cost function related to these tasks. We re-compute the MILP solution if the following condition is verified

$$\mu \triangleq |C^+(t_{opt}) - C^+(t_{use})|/C^+(t_{opt}) > \mu_t \qquad (14)$$

where μ_t is a positive threshold, $C^+(t_{opt})$ represents the cost function related to the tasks of interest according to the optimal solution computed at time t_{opt}, while $C^+(t_{use})$ is the cost function related to the same tasks according to solution in use computed at time t_{use}. The idea behind the condition in (14) is that a significant difference between the quantities $C^+(t_{opt})$ and $C^+(t_{use})$ can be indicative of the fact that the quality of the solution in use might be compromised since it has a cost significantly different compared to the last optimal one. Therefore, in this case, the optimization problem is solved again and the plan is updated accordingly. Note that for tasks already started, we enforce that they are allocated to the same service robots. The variable t_{opt} is finally set equal to t_{ch}.

VI. Simulation results

The validation of the proposed framework was executed within a virtual reality environment developed specifically for the project CANOPIES with Unity Engine, as shown in Figure 1, and reproducing a table grape vineyard. The considered vineyard consisted of 21 rows, with each row 3 m large and 15 m long. Regarding the agents, we considered four working agents ($n^a = 4$), comprising two humans ($n^h = 2$) and two robots ($n^w = 2$), and two service robots ($n^m = 2$). Specifically, the two working robots were composed of an Alitrak DCT-350P tracked mobile base and a PAL Robotics dual-arm system, while the two service robots were only given by the Alitrak tracked mobile base. Each working agent was equipped with a box initially empty and was required to perform $q_a = 3$ box filling operations $\forall a \in \mathcal{A}$, while the service robots were requested to provide box exchange services. The duration variables of these box filling operations were uniformly chosen in the interval $[250, 350]$ s. The box replacement times during service tasks δ^s_m and the box release times during depositing tasks δ^d_m were set to 10 s and 5 s, respectively, for all service robots, while the minimum $v_{\min,m}$ and maximum $v_{\max,m}$ velocities were selected as 0.2 m/s and 0.8 m/s, respectively, for all $m \in \mathcal{M}$. Regarding the working agent positions and the working agent base position, they were selected as shown in Figure 1. Finally, we set the weights $\alpha = \gamma = \kappa = 1$ and $\beta = 0.8$ and the re-allocation threshold $\mu_t = 0.1$. We utilized MATLAB interfaced with the Gurobi solver, to determine the MILP solution, and with ROS middleware, to command the robots and humans in the scene. A video showing the simulation results can be found in the supplementary material and at the link[1] (with higher resolution). Note that the box filling operations and the box exchange activities are not visualized in the simulation since they are part of the ongoing activities of the simulator development.

In Figure 2, the initial plan obtained by solving the MILP problem, reaching cost $c(t_{opt}) = 0.998$, is depicted on the left-hand side. The first two columns are associated to the service robots, the third and fourth columns to the humans, and the last two to the working robots. The time evolution is represented on the y axis. For the service robots, the thick lines represent the time from the start of a going task to the end of the service task, while the thin lines represent the depositing tasks. For the simplicity of representation, we do not make a distinction between going, waiting and servicing phases in the picture. For the working agents, the lines represent the times of the operations. The same colors are used to denote a working agent operation and the respective assistance tasks of a service robot. The figure shows that all the constraints for the correct execution of the tasks are fulfilled, e.g., each working agent waits for a service robot to pick up the previous box to start a new box filling operation. In addition, the robot velocities meet their allowed range (not shown in the figure for the sake of brevity) and the waiting times of the working agents are minimized. After starting the execution of the initial plan, we simulated that at time $t_{ch} = 300$ s a change in the human parameters occurred. Specifically, we considered that the second human moved by 10 m along his row in the vineyard in the direction opposite to the depot. This caused the length variables $l_{m,2}$ to change for all service robots. Therefore, according to the strategy in Section V, we first updated the plan by executing Algorithm 1, resulting in the plan shown in the middle of Figure 2. In the updated plan, we can observe higher waiting times of the working agents and makespan leading to an overall cost equal to $c(t_{use}) = 1.353$. This led the coefficient μ to overcome the threshold μ_t. Hence, the re-computation of the MILP solution was triggered as shown on the right of Figure 2, resulting in a reduced final cost equal to $c = 1.038$.

In addition, a simulation campaign was conducted to validate the proposed online updating strategy and study the effect of working agent variability on the optimality of the generated solutions. To this aim, we applied progressively higher variations to the duration variables of the working agents and analyzed the costs of the respective solutions. Specifically, we considered variations in the set $\mathcal{V} = \{60, 80, 100, 120, 140, 160, 180\}$ and, for each variation value ν, we applied perturbations distributed according to a uniform distribution $\mathcal{U}(\nu - 20, \nu + 20)$. The sign of the perturbation was obtained by uniformly selecting

[1] https://youtu.be/FU5PntUct7Y

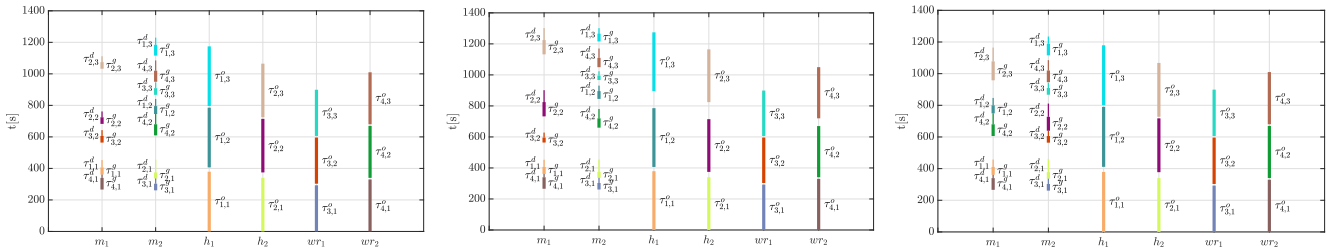

Fig. 2: Left: initial plan of the tasks on the left; middle: second plan updated at time t_{ch}; right: reallocation at time $t_{ch} = 300$ s.

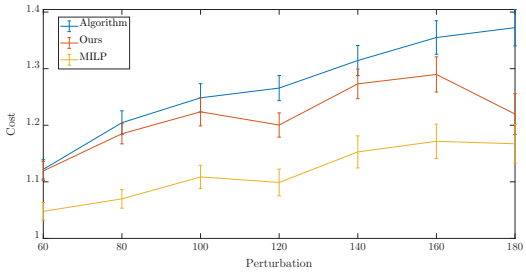

Fig. 3: Simulation campaign for perturbations of the working agent duration variables.

it between positive and negative. The change time was set to $t_{ch} = 250$ s, while the working agent with variations was randomly selected. Figure 3 shows the results obtained in terms of cost as the variation increases. Specifically, we compared the cost obtained by always re-computing the MILP solution (in red), by always running Algorithm 1 (in blue), and by executing our proposed updating strategy in Section V (in yellow). For each variation value, we applied 100 perturbations and reported the respective average and standard deviation of the cost. Obviously, we can observe that the cost obtained by always computing the MILP solution is the lowest since it corresponds to selecting the optimal solution at all times. Similarly, the cost obtained with only running Algorithm 1 is the highest since a very sub-optimal solution might be generated due to the shifting phases. Our approach instead leads to costs that fall in the middle between the above-mentioned two. In detail, for low perturbations, similar costs compared to the ones of the algorithm are found since the condition in (14) is not met; as the perturbations increase, we observe that our approach provides a greater gap from the algorithm cost since it performs reallocation in appropriate cases to preserve optimality. The figure thus shows the balance reached by our approach between the sub-optimality of the heuristics and the computational burden of the continuous MILP re-computation.

VII. Conclusion

We developed a framework for task allocation and scheduling in human-multi-robot settings where working agents, including working robots and human operators, perform operations in the environment and are assisted by service robots. A two-layer architecture was proposed: first, a MILP problem determines the allocation and scheduling variables for the services in order to minimize waiting times and energy consumption; next, an online updating strategy reacts to possible changes in the parameters of the working agents. A table-grape harvesting scenario in a precision agriculture context was considered to validate the proposed approach. As future work, we aim to make the cost function weights adaptive and validate the approach in a real agricultural field.

References

[1] A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib, "Progress and prospects of the human–robot collaboration," *Autonomous Robots*, vol. 42, pp. 957–975, 2018.

[2] S. E. Hashemi-Petroodi, S. Thevenin, S. Kovalev, and A. Dolgui, "Operations management issues in design and control of hybrid human-robot collaborative manufacturing systems: a survey," *Annual Reviews in Control*, vol. 49, pp. 264–276, 2020.

[3] C. L. Bethel, K. Salomon, R. R. Murphy, and J. L. Burke, "Survey of psychophysiology measurements applied to human-robot interaction," in *IEEE Int. Conf. Robot and Human Interactive Comm.*, 2007, pp. 732–737.

[4] C. A. Floudas and X. Lin, "Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications," *Annals of Operations Research*, vol. 139, pp. 131–162, 2005.

[5] S. Zhang, Y. Chen, J. Zhang, and Y. Jia, "Real-time adaptive assembly scheduling in human-multi-robot collaboration according to human capability," in *IEEE Int. Conf. Robot. Autom.*, 2020, pp. 3860–3866.

[6] M. Pearce, B. Mutlu, J. Shah, and R. Radwin, "Optimizing makespan and ergonomics in integrating collaborative robots into manufacturing processes," *IEEE Trans. Autom. Science and Engineering*, vol. 15, no. 4, pp. 1772–1784, 2018.

[7] M. Lippi, P. Di Lillo, and A. Marino, "A task allocation framework for human multi-robot collaborative settings," in *IEEE Int. Conf. Robot. Autom.*, 2023.

[8] A. Pupa and C. Secchi, "A safety-aware architecture for task scheduling and execution for human-robot collaboration," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, 2021, pp. 1895–1902.

[9] S. Alirezazadeh and L. A. Alexandre, "Dynamic task scheduling for human-robot collaboration," *IEEE Robot. Autom. Letters*, vol. 7, no. 4, pp. 8699–8704, 2022.

[10] A. Pupa, W. Van Dijk, C. Brekelmans, and C. Secchi, "A resilient and effective task scheduling approach for industrial human-robot collaboration," *Sensors*, vol. 22, no. 13, p. 4901, 2022.

[11] M. Zhang, C. Li, Y. Shang, and Z. Liu, "Cycle time and human fatigue minimization for human-robot collaborative assembly cell," *IEEE Robot. Autom. Letters*, vol. 7, no. 3, pp. 6147–6154, 2022.

[12] K. Li, Q. Liu, W. Xu, J. Liu, Z. Zhou, and H. Feng, "Sequence planning considering human fatigue for human-robot collaboration in disassembly," *Procedia CIRP*, vol. 83, pp. 95–104, 2019.

[13] T. Bänziger, A. Kunz, and K. Wegener, "Optimizing human–robot task allocation using a simulation tool based on standardized work descriptions," *J. Intell. Manuf.*, vol. 31, pp. 1635–1648, 2020.

[14] T. Yu, J. Huang, and Q. Chang, "Optimizing task scheduling in human-robot collaboration with deep multi-agent reinforcement learning," *J. Manuf. Syst.*, vol. 60, pp. 487–499, 2021.

[15] M. Lippi, J. Gallou, A. Gasparri, and A. Marino, "An optimal allocation and scheduling method in human-multi-robot precision agriculture settings," in *IEEE Mediterranean Conf. on Control and Autom.*, 2023.

[16] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. G. Lee, "A case study of mobile robot's energy consumption and conservation techniques," in *IEEE Int. Conf. Adv. Robot.*, 2005, pp. 492–497.

[17] R. Maderna, P. Lanfredini, A. M. Zanchettin, and P. Rocco, "Real-time monitoring of human task advancement," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst.*, 2019, pp. 433–440.

[18] G. Hoffman, "Evaluating fluency in human–robot collaboration," *IEEE Trans. Human-Mach. Syst.*, vol. 49, no. 3, pp. 209–218, 2019.

[19] M. Fischetti and A. Lodi, "Heuristics in mixed integer programming," *Wiley Encyclopedia of Oper. Res. and Manag. Science*, 2010.