

A Human-Centered Dynamic Scheduling Architecture for Collaborative Application

Andrea Pupa , Wietse Van Dijk , and Cristian Secchi 

Abstract—In collaborative robotic applications, human and robot have to work together during a whole shift for executing a sequence of jobs. The performance of the human robot team can be enhanced by scheduling the right tasks to the human and the robot. The scheduling should consider the task execution constraints, the variability in the task execution by the human, and the job quality of the human. Therefore, it is necessary to dynamically schedule the assigned tasks. In this letter, we propose a two-layered architecture for task allocation and scheduling in a collaborative cell. Job quality is explicitly considered during the allocation of the tasks and over a sequence of jobs. The tasks are dynamically scheduled based on the real time monitoring of the human’s activities. The effectiveness of the proposed architecture is experimentally validated.

Index Terms—Human-robot collaboration, Human-centered robotics, task planning.

I. INTRODUCTION

IN RECENT years, industrial setting has been supported by a constant increase in the use of collaborative robotics (see e.g., [1]).

The shift towards collaborative robotics can significantly change the quality of the job for the human. In fact, collaborative robots can take over dull, heavy or dangerous tasks making the life of the human easier. To ensure that the re-distribution of tasks is favorable to the human, the job quality aspect has to be taken into account when dividing tasks between human and robot [2]. This process can be guided by using job-quality metrics. The job quality framework, as used by the OECD [3], is a multidimensional concept that covers various topics. The part of job quality that concerns the quality of the working environment (e.g., time pressure, physical risks and work autonomy) is of specific interest for human-robot collaboration. At an even lower level, aspects of task load are governed by guidelines and

regulations, for example for lifting loads [4] and noise exposure [5].

In the industrial scenarios, collaborative cells are built in order to enable the human and the robot to work together on various jobs, each of which composed by a set of tasks. The distribution of task determines the load to which the human is subjected ([2], [6]). Additionally, the distribution of tasks determines influences the fluency, which in turn is strongly related to job quality aspects [7].

A lot of research has been done in the multi-agent task allocation problem in the industrial cases (see e.g., [8]–[10]). In general, these solutions cannot be directly applied in a human-robot collaboration (HRC) application, as they consider the presence of homogeneous agents.

Task allocation for collaborative cells has been modeled as a nonlinear optimization problem (see e.g., [11]–[14]) but the computational complexity of the problem is often high and it does not explicitly allow to take into account variable job-quality parameters. In [15] a two-level feedforward optimization strategy for offline subtask allocation between human and robot is presented. This strategy is integrated with a feedback procedure based on mutual trust to re-allocate the subtasks online. In [16] the authors propose a multi-criteria decision-making framework for task allocation and online re-scheduling which generates a solution that best matches the criteria you want to optimize. The re-scheduling procedure, however, happens only in case of unexpected events, without directly considering the real tasks duration. In [17] a two-level framework for task assignment is presented that dynamically handles task failures in a HRC scenario. Nevertheless, job quality over several jobs is not considered. Task assignment strategies for HRC assume that both the human and the robot require a constant amount of time to perform a task. This assumption may lead to inefficiencies. In reality, the human does not always take the same amount of time to accomplish the same task. Several works that consider task rescheduling with a variable human execution time are available in the literature (see e.g., [18], [19]) but the operator and the robot are treated as two separate entities and human-robot interaction and communication is not considered.

In order to make the human-robot collaboration as natural as possible, it is necessary to improve mutual awareness and communication both at the task execution level and at the task planning and scheduling level. Thus, it is important to have a monitoring strategy that makes the scheduler aware of the real duration of the tasks executed by the human and about the job quality of the human. This strategy allows the scheduler to adapt

Manuscript received October 15, 2020; accepted February 28, 2021. Date of publication March 25, 2021; date of current version April 13, 2021. This letter was recommended for publication by Associate Editor F. Mastrogiovanni and Editor G. Venture upon evaluation of the reviewers’ comments. This work was supported by the European Union’s Horizon 2020 Research and Innovation Programme under Grant Agreement 818087 (ROSSINI). (*Corresponding author: Andrea Pupa.*)

Andrea Pupa and Cristian Secchi are with the Department of Science and Method of Engineering, University of Modena and Reggio Emilia, Reggio Emilia 42124 RE, Italy (e-mail: andrea.pupa@unimore.it; cristian.secchi@unimore.it).

Wietse Van Dijk is with the Netherlands Organisation for Applied Scientific Research - TNO, Delft NL-2628, The Netherlands (e-mail: wietse.vandijk@tno.nl).

This letter has supplementary downloadable material available at <https://doi.org/10.1109/LRA.2021.3068888>, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3068888

the assigned tasks improving the job quality and the efficiency of the human-robot collaboration. Moreover, both the human and the robot should be able to communicate through the scheduler in order to improve the collaboration and to make it as natural as possible. The human, due to its expertise and experience, should be able to decide to execute a task that was previously assigned to the robot. While the robot should be able to assign a specific task to the human, if some failure occurs (e.g., a robot tool broke). All these decisions should be handled by the scheduler which re-assigns the tasks accordingly.

In this letter we propose novel framework for task assignment and scheduling for collaborative cells that is aware about the activity of the human and that allows the human and the robot to take decisions about the tasks they need to execute. This framework intrinsically considers job quality in the scheduling algorithm, in order to improve job quality of human workers. The proposed framework is composed by two layers. The first layer assigns, off-line, the tasks within a job to either the human or the robot by providing a nominal schedule. It considers the actual job quality indexes and the dependencies between the tasks. The second layer, the scheduler layer, reschedules the tasks considering the real execution time of the human operator, if needed, and the decisions taken online both by the human and the robot.

The main contributions of this letter are:

- A novel adaptive framework for task assignment and scheduling that considers into account real execution time, the job quality of the human, and the communication with human and robot for dynamic rescheduling
- A strategy for dynamic rescheduling that is effective and computationally cheap, i.e. suitable for industrial applications, and that allows human and robot to communicate their needs to the scheduler.

The letter is organized as follows: Section II presents the task assignment and dynamic scheduling problem for a collaborative cell. Section III presents the overall proposed architecture. Section IV an optimization problem for solving the task assignment and scheduling considering job quality is proposed. Section V defines the quantitative job quality metrics and attractiveness factor for human-robot collaboration. Section VI presents an algorithm that dynamically schedules the tasks for the human and the robot. Section VII summarizes the experimental validation of the proposed architecture in a custom HRC assembly scenario. Section VIII sums the conclusions and proposes future work.

II. PROBLEM STATEMENT

A collaborative industrial workspace is characterized by the presence of two different agents, a human operator H and a robot R , that must cooperate during a work shift in order to perform S jobs (J_1, \dots, J_S) . Each job consists out of one or more tasks¹ (T_{1j}, \dots, T_{Nj}) , each of which is characterized by an intrinsic cost w_{Ai} and by a *nominal* execution time t_{ai} , where $a \in A = \{H, R\}$ represents the agent that executes the task i . In the rest of the letter the double index is removed for ease of notation and

¹The choice of the specific technique for splitting a job into several tasks is out of the scope of this letter. Several strategies are available in the literature (see, e.g., [17] for assembly tasks.)

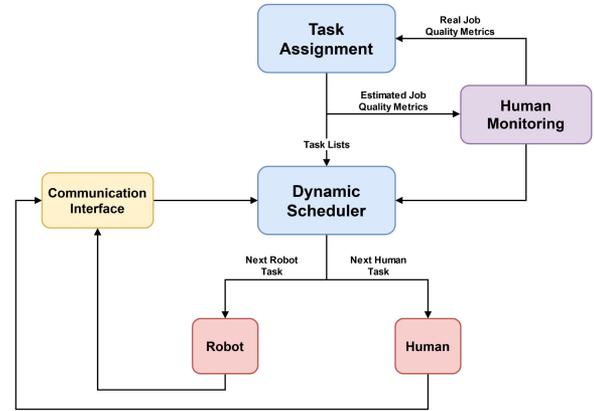


Fig. 1. The overall architecture. The blue blocks represent the two layers. The yellow and the purple blocks symbolize the strategies implemented to provide richer information to the architecture. The red blocks represent the two agents.

we refer to the set of tasks (T_{1j}, \dots, T_{Nj}) as (T_1, \dots, T_N) . The real task execution time can differ from the nominal execution time, due to uncertainties in the human behavior. Therefore, the workspace is equipped with a monitoring unit that, for each task T_i assigned to the human, estimates online the real execution time. To achieve this, many solutions can be found in literature, e.g., sequential interval networks [20], interaction probabilistic movement primitives [21], Open-Ended Dynamic Time Warping (OE-DTW) [22]. The output from the monitoring unit is used to update all relevant parameters over the entire work-shift.

The tasks composing a job may depend on each other, i.e. there could be precedence constraints, and the execution order must be considered when assigning tasks.

In this work, we aim at designing a task assignment and dynamic scheduling architecture that:

- Builds optimal nominal task schedules for the human and the robot, i.e. two task schedules such that, considering the nominal execution times, the precedence constraints, and job quality metrics, minimizes the job makespan, maximizing the parallelism between human and robot (i.e. minimizes waiting time), and optimizing the job quality for the human operator over the entire work shift.
- Starting from the nominal task schedules, reschedules both the human and the robot tasks according to the real execution time detected by the monitoring unit and the decisions taken by the human and the robot for task swapping. The rescheduling aims at minimizing the job makespan and improving the collaboration between the two agents.

III. ARCHITECTURE

The proposed task assignment and dynamic scheduling strategy is shown in Fig. 1, where two main layers can be distinguished:

- 1) **The Task Assignment Layer** is responsible of generating initial nominal schedules for the robot and the human, based on the maximum parallelism criterion, taking into account precedence constraints and job quality metrics over the entire work shift.

- 2) **The Dynamic Scheduler Layer** is responsible of scheduling the tasks, considering the real execution time and the requests coming from the human and from the robot.

Once the Task Assignment layer computes the initial nominal schedules, an estimation of the human job quality parameters is performed. This estimation is based on the nominal execution time t_{Hi} .

The resulting estimate represents the expected job quality parameters if the behavior of the human was exactly as the nominal one. To accommodate for (expected) deviations from the nominal schedule, the estimated job quality parameters are given as input to the **Human Monitoring** block, which is responsible for supporting and improving the scheduling procedure.

The Human Monitoring block aims to track the real execution time of the human operator during the execution of assigned tasks. The information about the real human behavior is then leveraged to update the estimated parameters. Subsequently, the real parameters that come out of the Human Monitoring block are used as input for the Task Assignment layer when calculating the nominal schedules of the new job. This feedback procedure allows to keep track of the evolution of job quality throughout the entire work shift, adapting each schedule accordingly.

The real execution time of the human is also exploited by the Dynamic Scheduler, which aims to reschedule, in real-time, the nominal tasks schedules. Frequently changing the order of the tasks assigned to the human can lead to confusion and poor efficiency of the human [23]. Thus, we have chosen to focus the rescheduling strategy primarily robot tasks. The list of tasks assigned to the human changes only when required by the **Communication Interface** block, namely when the robot cannot execute a task and a failure occurs, and when the human decides to perform a task instead of the robot. In all these cases, the changes in the human schedule are necessary and minimal.

IV. TASK ASSIGNMENT

The role of the Task Assignment layer is to build, for each job, the nominal task schedules for the human and the robot, taking into account job quality and precedence constraints. This relation of dependency between tasks can be represented with a directed acyclic graph $G = (T, E)$, as shown in Fig 2(a). Each vertex represents a task T_i while each directed edge E_{ij} means that the task T_i must be executed before the task T_j . Some tasks could be independent of each other, since there is not a path that goes from one task to the other (e.g., T_1 and T_3). The graph can then be rearranged so that all the parallel tasks are grouped together into several sets called levels L_l , as shown in Fig. 2(b). The choice of how the tasks are assigned to each level has a large impact on the schedules.

In this letter, both the problem of allocating the tasks to each agent and the way in which the tasks are distributed over the levels are addressed by solving the following multi-objective Mixed Integer Linear Program:

$$\min_{x, c_l} \sum_{l=1}^L \sum_{i=1}^N (w_{Ri} x_{Ril} + w_{Hi} x_{Hil}) + \frac{1}{t_{A, \max}} \sum_{l=1}^L c_l$$

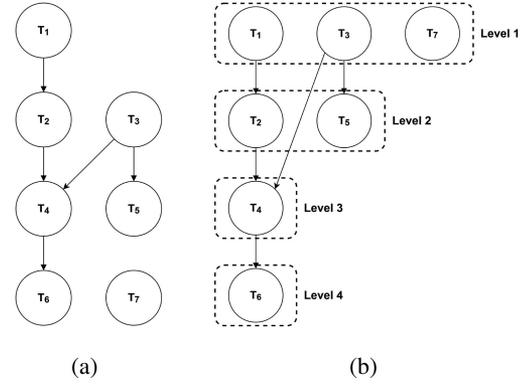


Fig. 2. Fig. 2(a) shows the directed acyclic graph of a Job composed by seven tasks, while Fig. 2(b) shows the division into four levels.

subject to

$$\begin{aligned} \sum_{l=1}^L (x_{Ril} + x_{Hil}) &= 1 \quad \forall i \in \{1, \dots, N\} \\ \sum_{i=1}^N t_{ai} x_{ail} &\leq c_l \quad \forall l \in \{1, \dots, L\}, \forall a \in A \\ \sum_{a \in A} \sum_{l=1}^L l \cdot x_{ail} &< \sum_{a \in A} \sum_{l=1}^L l \cdot x_{ajl} \quad \forall i \rightarrow j \\ K_m &\leq K_{m, \max} \quad \forall m \in \{1, \dots, M\} \\ K_{m, av} &\leq K_{m, av, \max} \quad \forall m \in \{1, \dots, M\} \end{aligned} \quad (1)$$

The terms $w_{Ri}, w_{Hi} > 0$ represent the weights for executing task T_i on behalf of the robot and of the human, respectively. The Boolean variables $x_{Ril}, x_{Hil} \in \{0, 1\}$ are detecting whether T_i is assigned or not to the robot or to the human, respectively, and at what level it must be executed; $x = (x_{R11}, \dots, x_{RN L}, x_{H11}, \dots, x_{HN L})$ is the vector containing all the decision variables. $t_{ai} > 0$ represents the nominal execution time of T_i on behalf of agent $a \in A$ and $t_{A, \max}$ is the maximum task duration. $c_l > 0$ denotes the cycle time of the l^{th} level and K_m and $K_{m, av}$ are quantitative parameters used to evaluate the m^{th} job quality metric, as detailed in Section V.

w_{Ri} and w_{Hi} are exploited for encoding the cost required by each agent to perform the task (e.g electrical cost, tool wear, or risk assessment). Very high weights are exploited for communicating to the task assignment algorithm that an agent is unsuitable for the execution of a task. Moreover, the human weights are exploited also to embed and evaluate in a quantitative way the job quality. The calculation method of these costs is a design parameter (see e.g., [24], [25]). In a general way we can define $w_{Ri} = h(costs)$ and $w_{Hi} = g(costs, job\ quality)$.

The first constraint guarantees that each task is assigned either to the robot or to the human. The second constraint maximizes the parallelism between the human and the robot. In fact, since all the terms in the quantity to minimize are positive, the optimization problem would tend to choose $c = \sum_{i=1}^L c_l$ as small as possible and the lower bound for this sum is given by the

third constraint and corresponds to the maximum parallelization of the activities of the human and of the robot. The third constraint ensures the respect of the precedence relationship, since all the tasks that should be executed before another are assigned to an upper level. The last constraints impose that the job quality metrics for the human operator will not violate the upper bounds.

The outcome of the optimization problem (1) are S_H and S_R , the set of tasks that have to be executed by the human and by the robot, respectively, and the level at which the task must be executed. This generates the nominal schedules, i.e. two ordered tuples S_H and S_R containing the tasks that have to be sequentially executed by each agent in each level.

V. JOB QUALITY METRICS FOR HRC

Job quality is considered in the Task Assignment via two mechanisms. The first mechanism ensures that various metrics related to job-quality do not exceed threshold values via optimization constraints. The second mechanism considers the overall attractiveness of the task-set that is assigned to the human. The attractiveness of the task-set is made part of the optimization objective.

Various metrics can be defined that describe a certain task load, e.g., the weight that is lifted, or the noise that is experienced during task execution. To monitor these metrics each task is assigned a set of weights (k_{i1}, \dots, k_{iM}). The metrics (K_1, \dots, K_M) can be calculated using two different generic representation of a job quality metric:

- *summed weight*:

$$K_m = K_{m,0} + \sum_{l=1}^L \sum_{i=1}^N (x_{Hil} k_{im}) \quad (2)$$

- *average weight*:

$$K_{m,av} = \frac{t_e K_{m,0} + \sum_{l=1}^L \sum_{i=1}^N (x_{Hil} t_{Hi} k_{im})}{t_m + c} \quad (3)$$

Where $c = \sum_{l=1}^L c_l$ is the cycle time. Typically, job-quality metrics are calculated over a time span longer than the execution of one task, t_m is the elapsed time of the time frame that is relevant for the metric, e.g., elapsed time since the human started the work shift. $K_{m,0}$ is the cumulative costs from previous jobs within the time frame t_e . The set of cumulative costs of each metric represents the job quality metrics that come out from the Human Monitoring block as shown in Fig. 1. It is worth noting that this cumulative cost guarantees that all the desired job quality metrics are estimated and constrained over the relevant time frames (t_m) and not just only over the single schedule.

Ensuring that pre-set thresholds on job-quality aspects are not exceeded does not automatically assign the tasks that are preferred by the human to the human. This aspect is governed by a general attractiveness factor that is assigned to each task. The attractiveness of the tasks is included in the optimization problem (1), i.e. in the evaluation of w_{HI} , so the scheduler tries to assign those tasks to the human, that the human likes to perform the most.

VI. DYNAMIC SCHEDULER

Starting from the output of the Task Assignment, the goal of the Dynamic Scheduler is to adapt online the two nominal schedules S_H and S_R taking into account the uncertainty of the human behavior. When two humans collaborate, their natural synergy allows them to reach high team performance. If one human gets slower, the other can compensate by speeding up. Furthermore, more complex unexpected difficulties are handled by communication. Experienced team members can exploit their knowledge to reorganize the work or decide to take over a difficult task. Less experienced team members can ask the expert member for some help when problems occur. The dynamic scheduler aims at reproducing this kind of behavior in human-robot collaboration in order to create an effective and natural cooperation.

This is achieved exploiting two different strategies. Firstly, the human operator is monitored in real time when performing the task in order to estimate the real execution time and, if necessary, to reschedule the future activities of the robot, reducing waiting time. Secondly, the communication between the human and the robot is enabled, allowing the two agents to take decisions about their activities through the scheduler. In particular, the robot delegates a task it cannot momentarily execute to the human. The human, instead, can decide to execute the task that the robot is performing (because, e.g., from its experience, it knows that the robot is not executing the task properly or to speed up the workflow). Moreover, the human can decide to re-assign some of its tasks to the robot. The dynamic scheduler is implemented according to the pseudo-code reported in Algorithm 1.

The dynamic scheduler needs as input the nominal task schedules S_H and S_R (Line 1). It firstly sets to false two variables End_R and End_H , which identify if the respective agent has concluded its task, and it initializes the job at the first level (Lines 2–3). Subsequently, if applicable, the algorithm assigns the first tasks of $S_R(l)$ and $S_H(l)$ to the human and to the robot (Lines 4–7). At this point, it starts two loops to check the end of the job (Line 8) and the actual level (Line 9), respectively. Inside the second loop, the scheduler first checks if the robot has performed all its tasks in the actual level. If this is true, the robot is in idle, waiting for the human to finish its task, and some tasks may be rescheduled (Line 11), maximizing the parallelism between the two agents. In the other cases the algorithm exploits the function $\mathbf{monitor}(T_R)$ to check if the robot has finished the assigned task (Line 13). The $\mathbf{monitor}()$ function can be implemented using standard procedures, available for robotic applications (see e.g., [26]). If the robot cannot succeed to execute T_R (e.g., a timeout error), a *delegate* message M_R is communicated. Subsequently, the algorithm checks if the human has completed its task, e.g., exploiting an HMI, and all the messages generated by the human and the robot are considered for task swapping (Lines 17). Afterwards, the algorithm checks if the two agents have concluded their tasks and, if it is the case, assigns them the next task in the level (Lines 18, 20). If no tasks are scheduled in the actual level, the function $\mathbf{next}(T, S(l))$ returns \emptyset . When both T_R and T_H are empty, then the level is concluded and the job moves on to the next one (Line 23). Finally, when

Algorithm 1: DynamicScheduler().

```

1: Require:  $S_H, S_R$ 
2:  $End_R, End_H \leftarrow false$ 
3:  $l \leftarrow 1$ 
4: if  $S_R(l) \neq \emptyset$  then  $T_R \leftarrow S_R(l, 1)$ 
5: end if
6: if  $S_H(l) \neq \emptyset$  then  $T_H \leftarrow S_H(l, 1)$ 
7: end if
8: while  $l \leq L$  do
9:   while ( $T_R \neq \emptyset$  and  $T_H \neq \emptyset$ ) do
10:    if  $T_R = \emptyset$  then
11:       $S_R \leftarrow \text{reschedule}(T_H, S_R)$ 
12:    else
13:       $End_R \leftarrow \text{monitorR}(T_R)$ 
14:    end if
15:     $End_H \leftarrow \text{checkEndH}()$ 
16:     $M_H \leftarrow \text{readH}(), M_R \leftarrow \text{readR}()$ 
17:     $(End_H, End_R, S_H, S_R) =$ 
    communication( $M_H, M_R, S_H, S_R$ )
18:    if  $End_H$  then  $T_H \leftarrow \text{next}(T_H, S_H(l))$ 
19:    end if
20:    if  $End_R$  then  $T_R \leftarrow \text{next}(T_R, S_R(l))$ 
21:    end if
22:  end while
23:   $l \leftarrow l + 1$ 
24: end while
25: updateJQ()

```

Algorithm 2: Reschedule().

```

1: Require:  $T_H, S_R$ 
2:  $t_{res} \leftarrow \text{monitorH}(T_H)$ 
3: if  $t_{res} > t_{Ri}$  then
4:    $(pS_R, fS_R) \leftarrow \text{split}(T_R, S_R)$ 
5:    $fS_R^r \leftarrow \text{fill}(fS_R, t_{res} - t_{Ri})$ 
6:    $S_R \leftarrow \text{concat}(pS_R, S_R^r, fS_R/fS_R^r)$ 
7: end if
8: return  $S_R$ 

```

all the tasks have been performed, the job quality parameters are updated through the function **updateJQ**() (Line 25) and used as input for the Task Assignment of the next job.

The rescheduling algorithm is represented in Algorithm 2.

The algorithm requires as input the task T_H is currently assigned to the human and the current robot schedule S_R (Line 1). It exploits the human monitoring strategy to estimate the remaining time t_{res} for the accomplishment of T_H (Line 2). The procedure **monitorH** can be implemented using several strategies available in the literature as, e.g., [20]–[22]. If t_{res} is greater than the time necessary for the execution of some tasks in S_R , then these tasks may be executed in parallel with T_H and, therefore, the rescheduling procedure starts (Line 3). First, the schedule S_R is split into two sub-lists: pS_R contains all the robot assigned tasks in the actual level while fS_R contains all the tasks of the next levels, which still need to be performed (Line 4).

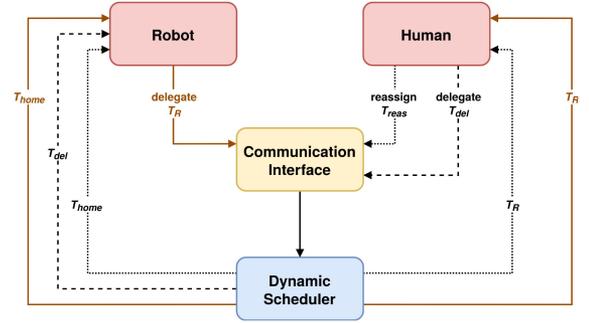


Fig. 3. Communication Layout. The dotted lines indicate the “reassign” message coming from the human and the consequent task scheduling. The dashed lines indicate the “delegate” message coming from the human with the following scheduling to the robot. Finally, the brown lines indicate the “delegate” message communicated by the robot.

fS_R is then used to create another sub-list S_R^r , which contains all the tasks that can be executed in the extra time available $t_{res} - t_{Ri}$ and whose precedences have already been executed (Line 4). Subsequently, $S_R(l)$ is updated by concatenating pS_R , with the list of the rescheduled tasks fS_R^r (Line 6). Finally, the new schedule S_R is returned.

During the execution of the job the human and the robot can generate messages in order to communicate to the Dynamic Scheduler the intention or need to swap their tasks. A detailed representation of this communication layout is shown in Fig. 3. In particular, the message M_R sent by the robot can be either empty or containing the value “delegate T_R ” and it is generated by the **monitorR** function when the robot cannot succeed in executing the assigned task. The message M_H can be either empty or it can assume two values: “reassign T_{reas} ” or “delegate T_{del} ”. The first message is generated when the human decides to execute the task that the robot is executing (because, e.g., the robot is not doing the assigned work properly or in the best way). The second message is generated when the human decides to delegate some tasks in S_H to the robot. This message has an argument, that specifies the task to be delegated. The human can enter the messages through a proper, job dependent, input interface. The messages generated by the human and by the robot are handled by Algorithm 3.

The algorithm requires the messages generated by human and robot M_R and M_H , the current schedules S_R and S_H and the task T_R currently assigned to the robot (Line 1). The message M_H is the first to be handled in order to give priority to the decisions taken by the operator. If the human decides to execute a task that was initially assigned to robot, it is necessary to check if the robot already started this task. If this is true, the robot task execution is aborted, moving the robot in an home safe position. T_{reas} is then deleted from S_R and pushed in the first position of the human schedule (Lines 3–9). The human can also decide to delegate a task $T_{del} \in S_H$. If T_{del} is executable by the robot, it is deleted from S_H and transferred into the robot schedule (Lines 10–13). The robot, instead, could detect that it cannot fulfill the assigned task and, if the task is executable by the human operator, T_R is deleted from the robot schedule and inserted in the schedule of the human, while a homing mission T_{home} is added as the next

Algorithm 3: Communication().

```

1: Require:  $M_H, M_R, T_R, S_H, S_R$ 
2: if  $M_H = \text{reassign}(T_{reas})$  then
3:    $End_H \leftarrow true$ 
4:   if  $T_{reas} = T_R$  then
5:      $End_R \leftarrow true$ 
6:      $S_R \leftarrow \text{push}(T_{home}, S_R)$ 
7:   end if
8:    $S_R \leftarrow \text{delete}(T_{reas}, S_R)$ 
9:    $S_H \leftarrow \text{push}(T_{reas}, S_H)$ 
10: else if  $M_H = \text{delegate}(T_{del})$  and  $\text{exRobot}(T_{del})$ 
then
11:    $End_H \leftarrow true$ 
12:    $S_H \leftarrow \text{delete}(T_{del}, S_H)$ 
13:    $S_R \leftarrow \text{push}(T_{del}, S_R)$ 
14: end if
15: if  $M_R = \text{delegate}(T_R)$  and  $\text{exHuman}(T_R)$  then
16:    $End_R \leftarrow true$ 
17:    $S_R \leftarrow \text{delete}(T_R, S_R)$ 
18:    $S_R \leftarrow \text{push}(T_{home}, S_R)$ 
19:    $S_H \leftarrow \text{push}(T_R, S_H)$ 
20: end if
21: return  $End_H, End_R, S_H, S_R$ 

```

task for the robot (Lines 15–19). Finally the procedure returns the updated end of task variables and schedules (Line 21). The procedures **exRobot** and **exHuman** exploit prior information about the job and the tasks (e.g., the weights w_{Ri} and w_{Hi} in (1)) to detect if a task can be executed by the robot or by the human. Note that the communication algorithm allows to manage only temporary errors. If the human perceives that a certain type of tasks should not be performed by the robot, e.g., the screwing tool broke, w_{Ri} must be adapted and the task assignment procedure must be re-executed.

Swapping the tasks between the human and the robot directly affects the job quality. Thus, it may happen that the final job quality indices do not respect the constraint imposed in the MILP problem (1). However, since the optimization problem requires as input the real job quality parameters, namely $K_{m,0}$, the possible work overload for the operator will be mitigated with subsequent task schedules. Another possible way to ensure the optimum of the job quality is the implementation of a function that checks if the task swapping will violate the constraints and, if required, prevents it. However the latter strategy is very conservative, as it does not take into account the possibility of correcting the job quality parameters in subsequent optimization problems. Furthermore, at the communication level, the job quality is treated as a soft constraint with respect to the need to swap a task (e.g., the robot failing a task is a more critical situation). For these reasons, in this letter it was decided not to investigate the latter solution.

The Dynamic Scheduler is computationally cheap. The heaviest part is represented by the **fill()** function inside the **Reschedule()** algorithm (see Alg. 2, Line 4). This function, in its worst implementation, has a linear complexity equal to



Fig. 4. Setup of the experiment. The two images show all the equipment used during the experiments.

$O(n + l)$, i.e. when analyzing every single element of the tuple with two *for* loops. Since the number of tasks and levels in an industrial application is not that large, the algorithm is reactive.

VII. EXPERIMENTS

The proposed two-layered framework has been experimentally validated in a custom collaborative assembly work shift, that has been set up only for evaluation purpose. During the experiments, the human operator cooperated with a UR10e, a 6-DoF collaborative robot. To monitor the human task execution we used a Kinect V2 RGB-D Camera with the official APIs for the skeleton tracking and to evaluate the remaining task time we implemented the OE-DTW algorithm, which is available in literature (see [27]), to both operator wrists. This algorithm returns the percentage completion of the task $\%_{compl}$, comparing the actual time series with the reference ones. This percentage was then exploited to estimate the remaining time of the task as $t_{res_i} = (1 - \%_{compl})t_{Hi}$. For the communication interface we used the computer keyboard to send signal through a simple HMI. The complete setup for the experiment is shown in Fig. 4.

All the software components were developed using ROS Melodic Morenia. The optimization problem was implemented using Python-MIP,² a collection of Python tools for the modeling and solution of Mixed-Integer Linear programs, and solved with Gurobi solver.³ The UR10e is position controlled using the ROS interface which accepts a desired final position while the overall trajectories are directly generated by the low level controller.

The work shift was composed by two jobs divided into multiple tasks. These are listed in Table I.

In the experiments we considered the average weight the human has to lift during the execution of the job $K_{1,av}$ as a job quality metric, with its respective upper bound $K_{1,av,max} = 1.1$.

The other inputs required by the Task Assignment Layer (see Section IV) are shown in Table II. The nominal durations were estimated by computing the average value of multiple measurements, while the intrinsic costs were calculated with the following equations:

$$w_{Ri} = 0.7D_{Ri} + 1000(1 - \text{capability}_i) \quad (4)$$

$$w_{Hi} = u_i \quad (5)$$

²<https://www.python-mip.com/>.

³<http://www.gurobi.com/>.

TABLE I
TASKS DESCRIPTION

Task Index	Description	Job
1	Pick&Place square shape.	J_1, J_2
2	Pick&Place U shape.	J_1, J_2
3	Pick&Place circular shape.	J_1, J_2
4	Pick&Place cross shape.	J_1, J_2
5	Pick&Place weight.	J_1, J_2
6	Pick&Place weight.	J_1, J_2
7	Packaging USB key.	J_1
8	Packaging USB key.	J_1
9	Packaging USB key.	J_1

TABLE II
TASK ASSIGNMENT DATA

Task Index	w_{Ri}	t_{Ri}	w_{Hi}	t_{Hi}	k_{i1}	$i \rightarrow j$
1	0.1	12	0.4	15	0	-
2	0.1	12	0.4	15	0	-
3	0.1	12	0.4	15	0	1, 2
4	0.1	12	0.4	15	0	1, 2
5	0.5	25	0.8	10	9	-
6	0.5	25	0.8	10	9	-
7	1000	-	0.4	25	0	-
8	1000	-	0.4	25	0	-
9	1000	-	0.4	25	0	-

where D_{Ri} represents the distance that the robot has to perform during the execution of the task T_i and $capability_i$ is a Boolean variable that indicates if the robot is capable of perform the task, e.g., the robot is not able to execute tasks T_7, T_8, T_9 . u_i is the attractiveness factor for the human in executing the task T_i and it was defined using our experience and knowledge in the work. k_{i1} are defined taking into account the weight of the objects, i.e. holding an USB stick or a shape does not affect the analyzed job quality metric. The only precedence constraints are related to the shapes and they are necessary to avoid the robot making a wrong pick with the magnets.

When the work shift starts, the job J_1 must be executed and the Task Assignment layer is initialized with the cumulative cost $K_{1,0} = 0$. The optimization problem is solved in 100 ms and the schedule is composed by the following two tuples:

- $S_H = \{[7, 8, 5], [9]\}$
- $S_R = \{[3, 4, 6], [1, 2]\}$

with an estimated job quality metric equals to:

$$K_{1,av} = \frac{t_{H5}k_{15}}{c} = 1.1 \quad (6)$$

Starting from the output of the Task Assignment Layer, the Dynamic Scheduler was then initialized and the two agents began to perform the collaborative job. A video of the experiments can be found in the supplementary material. The first part of the video is dedicated to the execution of the “nominal schedule”

of the job J_1 , i.e. the two agents perform exactly the assigned tasks. When the robot concludes all the tasks of the first level, the Dynamic Scheduler exploits the information coming from the OE-DTW to reschedule some tasks (see Alg. 2, Line 3–6). Since the monitoring algorithm returns an estimated remaining time $t_{res} > t_{R1}$, the robot can anticipate the task T_1 in the first level, instead of waiting for the level to finish as scheduled. This procedure is executed in 4 ms. It is worth noting that the estimation of the real execution time is the only variable that is evaluated at runtime, i.e. when the robot concludes the level. All the other metrics are evaluated after the execution of the entire job and used as input for the task assignment of the subsequent job.

Once J_1 is concluded the real execution time $t_{H5} = 15$ s and the real duty cycle $c = 79$ s are exploited to calculate the real job quality metric $K_{1,av} = 1.7$. The real job quality metric is then used as input for the assignment procedure of the next job J_2 . The new optimization problem is solved in 90 ms and the resulting schedules are:

- $S_H = \{[3], [1, 2]\}$
- $S_R = \{[4], [5, 6]\}$

It is worth noting that thanks to the job quality constraint no tasks affecting the weight metric are assigned to the human and the new estimated job quality metric is below the upper bound:

$$K_{1,av} = \frac{K_{m,0}}{t_e + c} = 0.96 \leq 1.1 \quad (7)$$

The second part of the video shows the execution of J_2 .

The communication strategy is then exploited. The work shift is reinitialized and the operator starts to execute the nominal schedule of the job J_1 . After concluding T_7 and T_8 he sends a message “delegate T_5 ” to the Dynamic Scheduler and the robot starts to execute this task instead of the human. At the end of the schedule, the human asks to the Dynamic Scheduler to reassign the task T_2 , and he performs that task instead of the robot. Since the human did not perform any task that affects the weight metric, the cumulative cost for the assignment of the J_2 is $K_{1,0} = 0$. For this reason, the output of the Task Assignment for J_2 is:

- $S_H = \{[], [6]\}$
- $S_R = \{[3, 4], [1, 2, 5]\}$

with an estimated weight metric $K_1 = 1$. This solution is obtained in 100 ms. It is worth noting that $c = 90$ s this is due to the fact that the Task Assignment schedules a pause before starting the execution of the second level. This pause is necessary to obtain an admissible value for the weight metric.

In order to demonstrate the effectiveness of the architecture, J_1 is performed without rescheduling the tasks. As happened before, while the human performs its tasks the robot concludes the first level as planned. Since the rescheduling is not active, the robot stops, waiting for the human to conclude the first level of the schedule. After the human executes T_5 , the schedule passes to the second level and two agents resume the expected behavior. As expected, the trial without the rescheduling takes more time ($c = 85$ s). Moreover, a great improvement can be seen in the robot idle times: $T_{R,idle} = 12$ s with our framework and $T_{R,idle} = 20$ s without using the rescheduling

procedure. It is worth noting that in both trials the human operator introduced approximately the same delay in the first level. The deviation from the nominal durations is in fact equal to $\mu_{proposed} = 1.67 \pm 6.2 s$ and $\mu_{baseline} = 1.33 \pm 4.7 s$. This means that the performance of the baseline architecture is not affected by further delays introduced by the human.

VIII. CONCLUSION

In this letter we have proposed a two layer architecture for dynamic task assignment and scheduling for collaborative application. Job quality has been explicitly considered. Rescheduling is performed considering real-time monitoring of the operator's activities and the human robot communication. The framework has been tested in a custom HRC assembly scenario and the results seem good, since the expectations have been satisfied.

Future work aims at validating the proposed architecture in an industrial scenario, where a real shift is considered and a user study is implemented. This will lead to a further validation of the architecture potential. Moreover, the HMI could be substituted with a different communication system that makes the message exchange simpler and more intuitive, while the OE-DTW could be replaced with a more reliable algorithm. Finally, how the job quality is affected by the rescheduling and the communication system could be analyzed, e.g., implementing a strategy that adapts online the attractiveness factor based on the past human preferences.

REFERENCES

- [1] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human-robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, 2018.
- [2] Q. Pham, R. Madhavan, L. Righetti, W. Smart, and R. Chatila, "The impact of robotics and automation on working conditions and employment," *IEEE Robot. Automat. Mag.*, vol. 25, no. 2, pp. 126–128, 2018.
- [3] Organisation for Economic Cooperation and Development, "Inventory for job quality," *Oecd Employment Labour Market Statist.*, 2020. [Online]. Available: <https://doi.org/10.1787/64b508c2-en>
- [4] NIOSH, "Work practices guide for manual lifting," Cincinnati, Ohio: US Department of Health and Human Services, *Public Health Service, CDC*, 1981, no. 81–122.
- [5] Council of European Union, "Directive 2003/ 10/ec," vol. 42, pp. 38–44, 2003. [Online]. Available: <https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX%3A32003L0010>
- [6] F. Wixted and L. O'Sullivan, "The effect of automated manufacturing environments on employee health," in *Proc. Irish Ergonomics Soc. Annu. Conf.*, vol. 1, 2014, pp. 80–91.
- [7] G. Hoffman, "Evaluating fluency in human-robot collaboration," *IEEE Trans. Human-Mach. Syst.*, vol. 49, no. 3, pp. 209–218, Jun. 2019.
- [8] C. Blum and C. Miralles, "On solving the assembly line worker assignment and balancing problem via beam search," *Comput. Operations Res.*, vol. 38, no. 1, pp. 328–339, 2011.
- [9] K. Xu, R. Fei, and D. He, "A tabu-search algorithm for scheduling jobs with precedence constraints on parallel machines," in *Proc. 13th IEEE Conf. Ind. Electron. Appl.*, 2018, pp. 2774–2781.
- [10] L. Sabattini, V. Digani, M. Lucchi, C. Secchi, and C. Fantuzzi, "Mission assignment for multi-vehicle systems in industrial environments," *IFAC-PapersOnLine*, vol. 48, no. 19, pp. 268–273, 2015.
- [11] G. Michalos, J. Spiliotopoulos, S. Makris, and G. Chryssolouris, "A method for planning human robot shared tasks," *CIRP J. Manuf. Sci. Technol.*, vol. 22, pp. 76–90, 2018.
- [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] A. Ayough, M. Zandieh, and F. Farhadi, "Balancing, sequencing, and job rotation scheduling of a u-shaped lean cell with dynamic operator performance," *Comput. Ind. Eng.*, vol. 143, 2020, Art. no. 106363.
- [14] K. Bogner, U. Pferschy, R. Unterberger, and H. Zeiner, "Optimised scheduling in human-robot collaboration-a use case in the assembly of printed circuit boards," *Int. J. Prod. Res.*, vol. 56, no. 16, pp. 5522–5540, 2018.
- [15] S. M. Rahman and Y. Wang, "Mutual trust-based subtask allocation for human-robot collaboration in flexible lightweight assembly in manufacturing," *Mechatronics*, vol. 54, pp. 94–109, 2018.
- [16] N. Nikolakis, N. Kousi, G. Michalos, and S. Makris, "Dynamic scheduling of shared human-robot manufacturing operations," *Procedia CIRP*, vol. 72, pp. 9–14, 2018.
- [17] L. Johannsmeier and S. Haddadin, "A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes," *IEEE Robot. Automat. Lett.*, vol. 2, no. 1, pp. 41–48, Jan. 2017.
- [18] P. Lou, Q. Liu, Z. Zhou, H. Wang, and S. X. Sun, "Multi-agent-based proactive-reactive scheduling for a job shop," *Int. J. Adv. Manuf. Technol.*, vol. 59, no. 1-4, pp. 311–324, 2012.
- [19] A. Casalino, A. M. Zanchettin, L. Piroddi, and P. Rocco, "Optimal scheduling of human-robot collaborative assembly operations with time petri nets," *IEEE Trans. Automat. Sci. Eng.*, vol. 18, no. 1, pp. 70–84, Jan. 2021.
- [20] N. N. Vo and A. F. Bobick, "From stochastic grammar to bayes network: Probabilistic parsing of complex activity," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 2641–2648.
- [21] G. Maeda, M. Ewerton, G. Neumann, R. Lioutikov, and J. Peters, "Phase estimation for fast action recognition and trajectory generation in human-robot collaboration," *Int. J. Robot. Res.*, vol. 36, no. 13-14, pp. 1579–1594, 2017.
- [22] R. Maderna, P. Lanfredini, A. M. Zanchettin, and P. Rocco, "Real-time monitoring of human task advancement," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 433–440.
- [23] S. P. Marshall, "The index of cognitive activity: Measuring cognitive workload," in *Proc. IEEE 7th Conf. Hum. Factors Power Plants*. 2002, pp. 7–7.
- [24] E. Lamon, A. De Franco, L. Peternel, and A. Ajoudani, "A capability-aware role allocation approach to industrial assembly tasks," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 3378–3385, Oct. 2019.
- [25] Z. Liu, X. Wang, Y. Cai, W. Xu, Q. Liu, Z. Zhou, and D. T. Pham, "Dynamic risk assessment and active response strategy for industrial human-robot collaboration," *Comput. Ind. Eng.*, vol. 141, 2020, Art. no. 106302.
- [26] O. Pettersson, "Execution monitoring in robotics: A survey," *Robot. Auton. Syst.*, vol. 53, no. 2, pp. 73–88, 2005.
- [27] P. Tormene, T. Giorgino, S. Quaglini, and M. Stefanelli, "Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation," *Artif. Intell. Med.*, vol. 45, no. 1, pp. 11–34, 2009.