A Human-Aware Method to Plan Complex Cooperative and Autonomous Tasks using Behavior Trees

Fabio Fusaro[1,2], Edoardo Lamon[1,3], Elena De Momi[2], and Arash Ajoudani[1]

***Abstract*— This paper proposes a novel human-aware method that generates robot plans for autonomous and human-robot cooperative tasks in industrial environments. We modify the standard Behavior Trees (BTs) formulation in order to take into account the action-related costs, and design suitable metrics and cost functions to account for the cooperation with a worker considering human availability, decisions, and ergonomics. The developed approach allows the robot to online adapt its plan to the human partner, by choosing the tasks that minimize the execution cost(s). Through simulations, we first tuned the weights of the cost function for a realistic scenario. Subsequently, the developed method is validated through a proof-of-concept experiment representing the boxing of 4 different objects. The results show that the proposed cost-based BTs, along with the defined costs, enable the robot to online react and plan new tasks according to the dynamic changes of the environment, in terms of human presence and intentions. Our results indicate that the proposed solution demonstrates high potential in increasing robot reactivity and flexibility while, at the same time, in optimizing the decision-making process according to human actions.**

I. Introduction

Nowadays, compliant lightweight robots are increasingly exploited to address the upcoming needs of the new industries, where cyber-physical systems continuously communicate and collaborate with each other and their human counterparts. Beyond the requirement for high flexibility to accomplish a wide variety of tasks, the improvement of worker ergonomics is of high relevance. Moreover, while achieving such tasks, robot behaviors should be adapted to the worker intentions and commands, from control to task planning level [1]. To address the above specifications, several attempts have been made to develop dexterous robots with agile motions, ranging from humanoids [2], [3], to single, or dual-arm wheeled manipulators [4], [5]. The latter category has received slightly more attention in industry due to an inherently greater postural stability, while dealing with heavy manipulation tasks. The examples include logistics [6]–[8] and manufacturing [9] scenarios. The mobility of such systems allows to exploit their loco-manipulation capabilities to ensure safe human-robot collaboration, through intuitive interfaces that allow the human operators to interact with the robot [10], [11]. Nevertheless, such interfaces should

1- HRI^2 Lab, Istituto Italiano di Tecnologia, Via Morego 30, 16163, Genova, Italy. 2- Department of Electronics, Information and Bioengineering, Politecnico di Milano, Milano, Italy. 3- Robotics and Automation, Dept. of Information Engineering, Universita' degli Studi di Pisa, Pisa, Italy. fabio.fusaro@iit.it

This work was supported in part by the European Research Council's (ERC) starting grant Ergo-Lean (GA 850932), and in part by European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 871237 (SOPHIA).

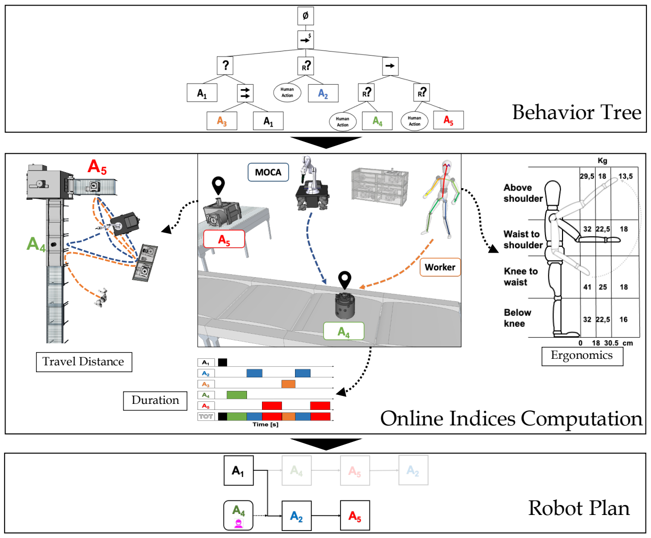


Fig. 1: Schematic overview of the proposed method. The reactive task planner, exploiting Behavior Trees (BTs), selects online the new robot task according to the minimization of a weighted cost, based on duration, ergonomics and travel distance indices.

not only allow to regulate the interaction at the control level but also adapt robot behavior to human intentions, actions and preferences.

In literature, different methods have been used to face the adaptation of robot behavior to human intentions or interactions. In auction-based planning, the plan is generated through human-robot communication using gestures [9] or verbal statements [12]. Alternative approaches, to address this problem, combine motion [13] to consider human activities together with dynamic changes of the environment [14]. For instance, a Human Aware Task Planner (HATP) for collaborative and interactive robotic applications producing *socially acceptable plans* for several agents is proposed in [15]. To anticipate human activities, in [16] the authors presented a learning method, based on Markov decision processes, that learns the human actions from RGB-D videos. One of the shortcomings of the approach is that the dataset is limited to human actions in domestic environments and the method requires re-training to be able to learn new tasks.

To solve the task planning problem, it is necessary to exploit models that allow taking into account all the possible robot actions and reactions in accomplishing a set or sequence of tasks. This can be achieved by taking advantage of heuristics [17] and deterministic [18] methods. Specifically, in [17], the authors proposed a method that can decompose a causal graph of a translated planning task into a sequence of tasks and to plan with a heuristic system. In [18], the presented approach generates plans including probabilistic values, which are obtained from Markov Logic Networks (MLN), a learning statistical relational model. The problem of blending decision making with task execution has been widely tackled with methods like Decision Trees (DTs) [19] and Hierarchical Finite State Machines (HFSMs) [20]. DTs are constituted by predicates and control statements that map possible consequences like event results and utility. HFSM is a FSM in which the states can be other FSMs. This structure is easy to implement and design and the hierarchical property reduces the state dimensions' growth when its complexity increases. Nevertheless, the state transitions have to be defined manually and cannot change dynamically, hence, each scenario-related behavior is usually not reusable.

Behavior Trees (BTs) represent another method, that has been extensively used in the behavior development of the Non-Player Character in the game AI. BTs can be considered as an evolution of HFSMs in which the states are replaced by atomic actions and the state transitions are defined implicitly in the BT structure [21]. Moreover, they have the advantages to be reactive, modular, maintainable and reusable. Recently, researchers started to develop methods to integrate task planning concepts with BTs. For instance, the CoSTAR system was developed to design robotic programming for non-expert users exploiting the user-friendly BTs structure [22]. The architecture allows users to plan abstract goals by defining generic actions. Extended BTs [23] were developed to generate a plan using a planner for a problem defined with the planning language Planning Domain Definition Language (PDDL), to create a hierarchical tree and then to optimize it to minimize the execution time and resources usage. In [24], the authors merged the PDDL with the Hierarchical task and motion Planning in the Now (HPN) to update dynamically the BT every time it fails. Similarly, in [25], the approach extended the BTs adding pre- and post-conditions. Finally, Utility BTs [26] allow creating behaviors where each action is selected among a set of actions, like the one that maximizes the utility value. The other actions are then discarded from the plan.

The standard BT structure and the above-mentioned extensions were not designed to operate in human-populated environments, and moreover in industrial scenarios. They do not provide the possibility to change online the structure of the BT. Furthermore, they do not envision the presence of other agents in the plan. In Human-Robot Collaboration, it is important not only to adapt to the dynamic changes of the environment but also to modify the generated plan with respect to human intentions, activities, motion and availability. To overcome these limitations, this paper proposes an online adaptation of the robot plan to human decision-making as a first step towards human-aware task planning using Behavior Trees. We extended Utility BTs in order to plan robot behaviors according to production-related indices, such as time performance, distance traveled by the agents and human ergonomics, ordering the sequences of actions to maximize the utility factor. One of the advantages of the proposed method is that the same BT can handle different levels of engagement between humans and robots: from *coexistence* to *cooperation*, and *autonomous* task execution. The developed method (see Figure 1) models the task-related cost considering distances from the human and the robot to the task, human ergonomics in terms of weights and object location, and the task duration. The performance of the proposed approach is evaluated experimentally both in simulation and in a real scenario, where the MObile Collaborative robotic Assistant (MOCA) plans each new task adapting and reacting to human presence while achieving object transportation tasks.

Type of Node	Symbol	Success	Failure	Running
Sequence	$\rightarrow$	All children succeed	One child fails	One child returns Running
Fallback	?	One child succeeds	All children fail	One child returns Running
Decorator	$\diamond$	Custom	Custom	Custom
Parallel	$\rightrightarrows$	$\geq M$ children succeed	$> N - M$ children fail	else
Condition	⬭	True	False	Never
Action	▭	Upon completion	Impossible to complete	During execution

TABLE I: BT node types and return status.

II. Behavior Trees for Human-Aware Task Planning

A. Preliminaries on Behavior Trees

A BT is a directed rooted tree, consisting of internal nodes for control flow and leaf nodes for action execution or condition evaluation. Pairs of adjacent nodes are denoted as *parent* and *child*. The only node without parents, the root, periodically sends a signal, called *tick*, through the tree, which is propagated to its children to allow their execution. Then, the queried child returns immediately a status to the parent, depending on the type of node: *SUCCESS* if the node successfully completed its execution, *FAILURE* if it failed and *RUNNING* if the execution is not complete. There are four standard types of control nodes (Fallback, Sequence, Decorator, Parallel) and two standard categories of execution nodes (Condition, Action). In Table I the standard node types are summarized, with their symbol and the return status depending on each case. A more detailed overview of standard BTs can be found in *Colledanchise et al.* [21].

B. Custom Cost Behavior Trees

First, we developed a custom decorator node to allow the execution of repetitive actions or sub-behaviors. The so-called *Keep Running Until Success* node ticks continuously the only child until it returns *SUCCESS*, so that, if an action fails, the action is repeated until it is achieved. The pseudocode of the *Keep Running Until Success* node, represented by the $\circlearrowright$ symbol, is synthesized in Algorithm 1.

Moreover, in the standard BT formulation, the execution order of sequence and selector node children is intrinsically fixed. Thus, the execution order of conditions and actions must be established beforehand by the programmer, which is, in the case of complex tasks, non-trivial. To enable adaptation to human presence we employ Utility BTs to assign a utility value to each action. This value is normally

Algorithm 1 Tick() function of the "KeepRunningUntilSuccess" node.

```
1: procedure KEEPRUNNINGUNTILSUCCESS::TICK()
2:     $child\_status \leftarrow child.Tick()$
3:     if $child\_status == RUNNING$ then
4:         return $RUNNING$
5:     else if $child\_status == SUCCESS$ then
6:         return $SUCCESS$
7:     end if
8: end procedure
```

used by the fallback node to pick the action that maximizes such value. We extend this concept, enabling a sequence node to order the actions according to the utility value. In this way, the following ticked child is no longer fixed but is selected online as the one with minimal cost. The pseudocode for a *Sequence Costs* node with N children, represented by the $\rightarrow^{\$}$ symbol, is summarized in Algorithm 2.

Algorithm 2 Tick() function of the "SequenceCosts" node.

```
1: procedure SEQUENCECOSTS::TICK()
2:     $costs_{idxs} \leftarrow$ SORT_INDICES($costs$)
3:     for $i \leftarrow 1$ to $N$ do
4:         while $\neg$ (FIND $costs_{idxs}(j)$ IN $executed\_child_{idxs}$) do
5:             $j{+}{+}$
6:         end while
7:         $idx \leftarrow costs_{idxs}(j)$
8:         $child\_status \leftarrow child(idx).Tick()$
9:         if $child\_status == RUNNING$ then
10:            return $RUNNING$
11:        else if $child\_status == FAILURE$ then
12:            CLEAR($executed\_child_{idxs}$)
13:        else if $child\_status == SUCCESS$ then
14:            ADD($idx$ IN $executed\_child_{idxs}$)
15:        end if
16:    end for
17:    CLEAR($executed\_child_{idxs}$)
18:    return $SUCCESS$
19: end procedure
```

C. Metrics and Cost Function Design

The meaning of the utility factor in game *AI* and also how it should change in relation to the situation, feelings, risks, etc., is quite clear [27], while, in human-populated environments, and especially in industrial ones, other factors influence the cost evaluations. We propose to use, as relevant factors in these scenarios, three different metrics that influence task performance: task execution time, human ergonomics, and total travel cost of each agent.

a) Duration Index: We consider the duration of a task as a suitable metric to measure the agent's performance in achieving the task. In manufacturing processes, such as assembly lines, the same sequence of tasks is continuously repeated. The average *task duration* represents a simple index to measure task performance. Moreover, this index could allow reducing the overall execution time in future executions. The duration index cost, T_i, is computed by:

$$T_i = \frac{\overline{\tau}_i - \tau_{i,des}}{\tau_{i,des}} \tag{1}$$

where $\overline{\tau}_i$ is the average execution time of the i-th task at the previous shift of the production line and $\tau_{i,des}$ is the nominal duration of the i-th task dictated by the manufacturing process. The higher the duration index, the slower the execution of the task. The minimization of such index allows executing first the actions the robot is able to achieve in time and the human does not. In this way, the robot will leave to the worker the tasks the latter is faster. Since the worker is not obliged to execute those tasks, the robot will schedule them as last. Moreover, if he is not present in the workcell, the robot will execute all tasks from the fastest to the slowest. In case the human enters later in the scene, the remaining tasks are the ones the robot performs slower than the previous ones.

b) Ergonomics Index: A fundamental aspect of the employment of cobots in industry is the opportunity to boost worker ergonomics while keeping productivity. Therefore, we would like to achieve that the robot entrusts itself with the heaviest and uncomfortable tasks. Thanks to ergonomics indicators, it is possible to assign to each task an execution-related cost that indicates the level of ergonomic risk of the worker in achieving such a task. In literature, different ergonomics indicators are present, from posture-related (*RULA*, *REBA*, *OWAS*) to more task-specific (*OCRA*, *NIOSH*, *EAWS*). In this work, we select an ergonomic risk assessment for manual material handling tasks, the Washington Industrial Safety and Health Act (*WISHA*) [28]. This index allows taking into account not only the weights of the objects but also the relative position of the task with respect to the human (in terms of height and horizontal distance), the frequency of the task execution in a day, the task duration and the body twist angle required to achieve the task. The cost related to the ergonomics index E_i is defined as:

$$E_i = \frac{u_i a_i l}{w_i} \tag{2}$$

where u_i is the unadjusted weight limit (i.e. the weight limit that a worker can lift, not considering the twisting adjustment) that changes according to vertical position and horizontal distance of the i-th task from the human, a_i is the twisting adjustment for the i-th task, l is the limit reduction multiplier that takes into account the frequency of repetition and the duration of tasks in a day and w_i is the weight of the i-th task[1]. For the sake of simplicity, in this paper, we consider the horizontal distance, a_i and l as constant and hence not affecting the computed value.

[1]The values of these variables are assigned using the tables in `https://ergo-plus.com/wisha-lifting-calculator-guide/`.

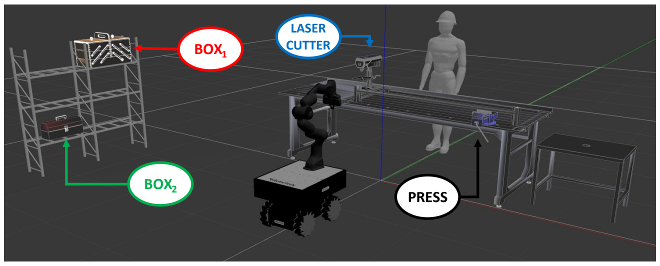


Fig. 2: Setup of the simulation experiments. The tasks on the workbench, starting from the right, are: press and laser cutter machines. The tasks on the shelves, starting from the top, are: box$_1$ and box$_2$.

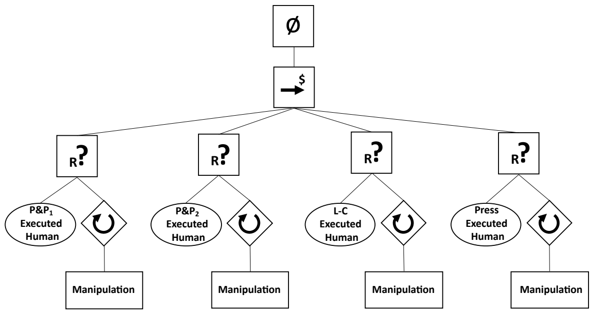


Fig. 3: BT of the simulation experiment. The $\rightarrow^{\$}$ is the "Sequence-Costs" node explained in Algorithm 2, the $_R?$ is the Reactive Fallback and the ↻ is the "KeepRunningUntilSuccess" node explained in Algorithm 1.

c) Travel Distance Index: We would like to account for the relative distance between tasks and agents. For instance, it is more likely that the human executes all the tasks close to his current position than he starts a new task far from him.For this reason, we consider a travel cost. In addition, we evaluate, with less priority, also a robot travel cost. In this way, among the tasks far from the human, the robot will pick the closest to the current one. Moreover, this component allows optimizing travel cost when the human is not present (and hence its travel cost is *0*). As travel cost we simply consider the Euclidean distance between task and robot position (R_i) and the inverse of the Euclidean distance between task and human (H_i), as:

$$H_i = \frac{1}{\|{}^h\boldsymbol{x}_{t_i}\|} \tag{3}$$

$$R_i = \|{}^r\boldsymbol{x}_{t_i}\| \tag{4}$$

where ${}^r\boldsymbol{x}_{t_i}$ and ${}^h\boldsymbol{x}_{t_i}$ are the distances between the human, the robot and the i-th task, respectively. In this way, each task close to the human will have a high cost, whereas a task close to the robot will have a lower cost.

d) Cost Function Design: To allow a meaningful comparison between the terms, each cost is normalized by dividing it by the maximum cost value of the same metric. Moreover, in the denominator of the computation of H_i and E_i, there is an additional $\epsilon > 0$, small enough, to avoid numerical issues. Then, the total cost of each task is computed as a weighted sum of the considered factors

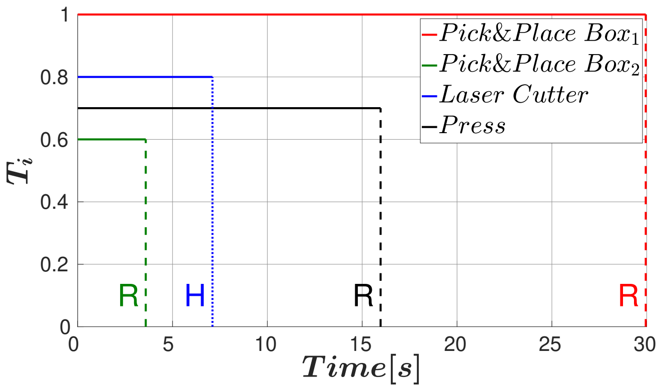


Fig. 4: Task duration costs of the 4 tasks of the simulation experiment. Weights: $\upsilon = 1$ and $\eta = \rho = \xi = 0$. In all cost plots, the dashed and dotted lines represent the instant in which the robot (R) and the human (H) choose the task to accomplish, respectively.

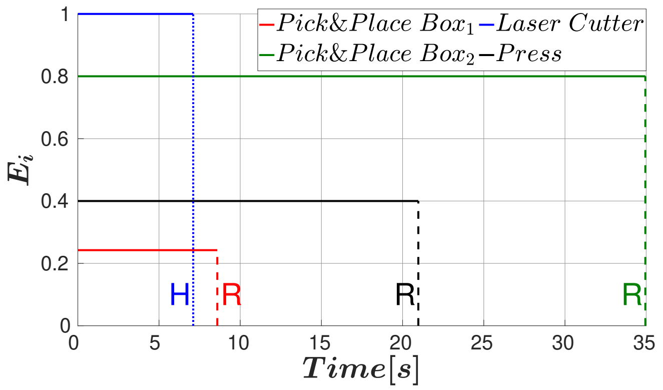


Fig. 5: Ergonomics costs of the 4 tasks of the simulation experiment. Weights: $\xi = 1$ and $\eta = \rho = \upsilon = 0$.

explained before. Hence, the total cost of the i-th task, C_i is simply defined as:

$$C_i = \eta H_i + \rho R_i + \xi E_i + \upsilon T_i \tag{5}$$

where η, ρ, ξ and υ are the semi-positive weights of the respective costs and can be tuned in order to obtain the desired behavior.

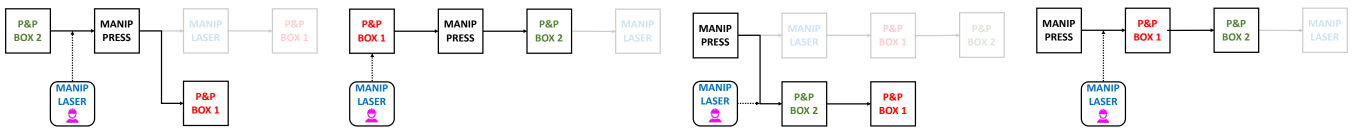
The cost weights can be set arbitrarily, but, in our scenario, we tuned them to obtain the desired robot behavior. For instance, the distance cost weights need to satisfy the following constraint:

$$\eta + \rho = 1, \tag{6}$$

The condition reflects the fact that human-task and robot-task distance are jointly normalized, and, if the human is not available, $\rho = 1$. Moreover, only if the human is present in the workcell, we assume $\eta > \rho$. This additional condition reflects the fact that if the two agents have the same distance from a task, and other tasks are available, the robot will leave such task to the human.

III. Simulation Experiments

First, we test the operation of the proposed approach in a simulated scenario. We consider 4 different tasks: the manipulation of an object, with two different tools, a press



(a) Robot plan computed with task duration cost in Figure 4.

(b) Robot plan computed with ergonomics cost in Figure 5.

(c) Robot plan computed with travel distance cost in Figure 7.

(d) Robot plan computed with the mixed cost in Figure 8.

Fig. 6: Plans of the simulated task. Each block represents a subtree of the BT in Figure 3. The non-transparent squared blocks represent the tasks accomplished by the robot. The transparent blocks are the planned tasks at the instant when the robot decided to accomplish the non-transparent one. The curved squared blocks are the tasks accomplished by the human. Each row shows the updated plan.

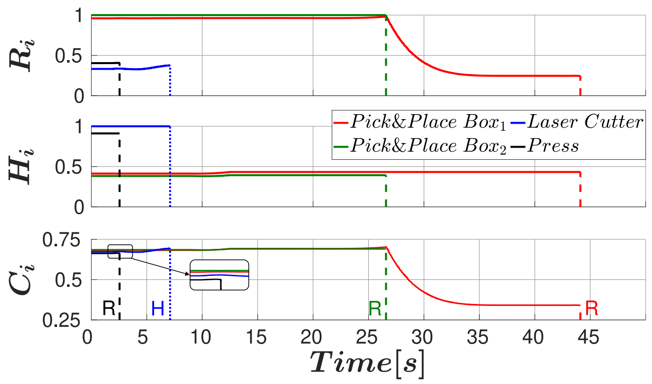


Fig. 7: Travel distance costs of the 4 tasks of the simulation experiment. Weights: $\eta = 0.51$, $\rho = 0.49$ and $\xi = \upsilon = 0$. The two plots in the top depict the cost considering only the contribution of the robot-task distance and the human-task distance, while the third plot shows the weighted sum of the two.

machine and a laser cutter, both located on the workbench in different positions, and the transportation of two toolboxes, located at different heights on the shelf. Moreover, we simulate the presence of a human worker.

a) Setup: The simulation setup can be found in Figure 2. In the simulation, the robot's goal is to accomplish all tasks. The robot is not aware of the human intentions and planned tasks, hence, it has to react according to worker action. In all the simulations, the objects' locations and the worker action are fixed. The worker always accomplishes the insertion of the object in the laser cutter at the same instant. The BT of the simulation is depicted in Figure 3. All the tasks can be either accomplished by the MOCA autonomously or cooperating with the human. The Reactive Fallback allows to continuously check the condition while executing the action, in order to react in case the worker decides to accomplish a task. The goal of the simulation is to verify the robot behavior according to the selected costs and weights. To do so, first, we test individually the proposed indices, then we combine them together by means of (5).

b) Results: First, the robot considers only the task duration in choosing which task to accomplish. The costs of the tasks in time are shown in Figure 4[2]. They are constant since the execution time does not change. The plan, in Figure 6a, is updated when the human accomplished the task using the laser cutter. Noteworthy, due to the normalization, the unit cost is always assigned to the most expensive task. The tasks are selected by the BTs starting with the most efficiently performed. Next, the weights are tuned in order to take into account only the ergonomics of the worker. So, the robot starts to plan from the most risky task for the human, which is the one with the lowest cost for the cobot, as shown in Figure 5. Also, in this case, the plan, in Figure 6b, is updated only when the human achieved the task related to the laser cutter. Next, the weights are tuned in order to consider only the human and robot distances to the tasks. The costs are shown in Figure 7 and the plan is illustrated in Figure 6c. The weights are chosen as: $\eta = 0.51$, $\rho = 0.49$ and $\xi = \upsilon = 0$. It can be noticed that the cobot starts to accomplish the farthest task from the human but, at the same time, closest to its position. Then the robot updates the plan when the human achieved the manipulation task with the laser cutter. Finally, the combined cost case is considered. The chosen weights are: $\eta = 0.51$, $\rho = 0.49$, $\xi = 0.6$ and $\upsilon = 0.35$. Since the human is present in the workcell, $\eta = 0.51$ and $\rho = 0.49$ are chosen to satisfy the conditions in (6), where the two weights are similar to not privilege a specific agent. Moreover, in our scenario, we would like to favor worker ergonomics instead of the execution time. For this reason, we select $\xi > \upsilon$. The plot of the costs in time can be seen in Figure 8. The plan, in Figure 6d, is not updated since the task of the laser cutter, later achieved by the human, is planned as last. Once the robot executed the transportation of the box$_2$, the plan is ended. Noteworthy, tuning the weights in different manners generates different plans and hence behaviors of the robot. Therefore, the design of the weights has a high impact on the cobot behavior and each user can design them in relation to the desired index to maximize.

IV. Experiments

The proposed approach is further validated in a proof-of-concept experiment with a fast-reconfigurable and flexible setup, inspired by the SOPHIA[3] project use-case.

Setup: A single cobot, MOCA, has to accomplish different tasks in sequence. MOCA is a robotics research platform made of a torque-controlled redundant manipulator mounted on top of a velocity-controlled mobile base. It is controlled by means of a Cartesian weighted whole-body impedance controller [7], [11] that executes smooth polynomial Cartesian trajectories. The depicted task is the

[2]For the sake of clarity, from now on, the dashed and dotted lines in all cost plots represent the instant in which the robot (R) and the human (H) choose the task to accomplish, respectively.

[3]Socio-Physical Interaction Skills for Cooperative Human-Robot Systems in Agile Production: `https://project-sophia.eu/`

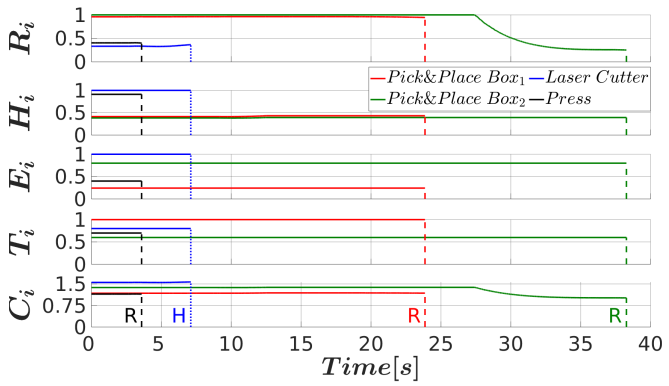


Fig. 8: Mixed costs of the 4 tasks of the simulation experiment. Weights: $\eta = 0.51$, $\rho = 0.49$, $\xi = 0.6$ and $\upsilon = 0.35$. The four plots in the top depict the cost considering the single contributions, while the fifth plot shows the weighted sum of the four.

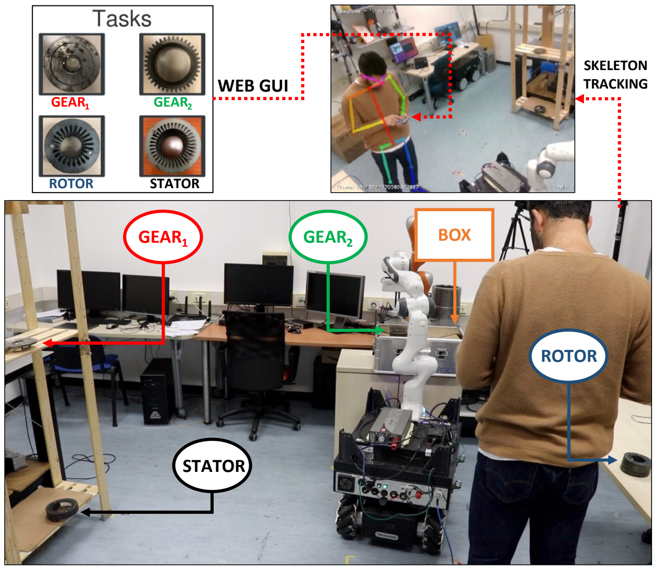
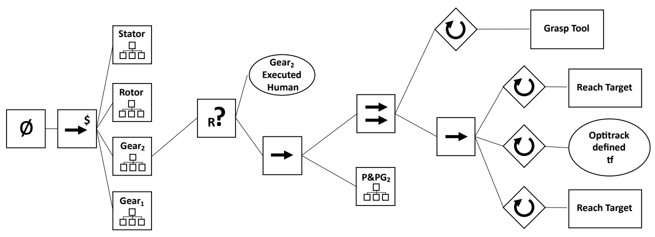
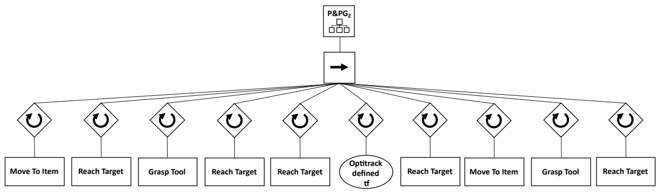


Fig. 9: Experimental setup. The human exploits a web GUI clicking on the phone screen the button of the task that he wants to accomplish. In the right of the image, a pole can be seen in which is attached a RGB-D camera that allows to track the skeleton of the worker. The objects are in the starting position, except the gear$_2$ which initial location is on the table next to the rotor. The goal location is the Box.

boxing of four parts of an electric motor, i.e., two gears, a rotor and a stator, with different dimensions and weights, located in different places in space. These 4 spots represent the end of the production line, where objects of the same type are manufactured, or different temporary containers where these objects are placed before being boxed and delivered. In our experiment, *gear*$_1$ and the *stator* are located on the shelf, the *rotor* and *gear*$_2$ on the table. The experimental setup is depicted in Figure 9. To track human pose and skeleton joint positions in real-time, we placed an RGB-D camera, an Intel RealSense D435i, fixed in the workcell, running OpenPose [29], similar to [11]. Thanks to the skeleton tracker, it was possible to account for human presence in the work-cell, automatizing the change of the cost weights. Moreover, the skeleton keypoints' positions were used to



(a) Horizontal representation of the BT of the experiment. The *tick* signal queries the children in top down order. The *Pick and Place gear*$_2$ (P&PG$_2$) subtree is illustrated in Figure 10b.



(b) *Pick and Place gear*$_2$ (P&PG$_2$) subtree of Figure 10a.

Fig. 10: BT of the experiments. The $\rightarrow^{\$}$ is the *SequenceCosts* node explained in Algorithm 2, the $_R?$ is the Reactive Fallback and the ↻ is the *KeepRunningUntilSuccess* node explained in Algorithm 1. *gear*$_1$, *rotor* and *stator* subtrees have the same structure of the *gear*$_2$ subtree showed in Figure 10a.

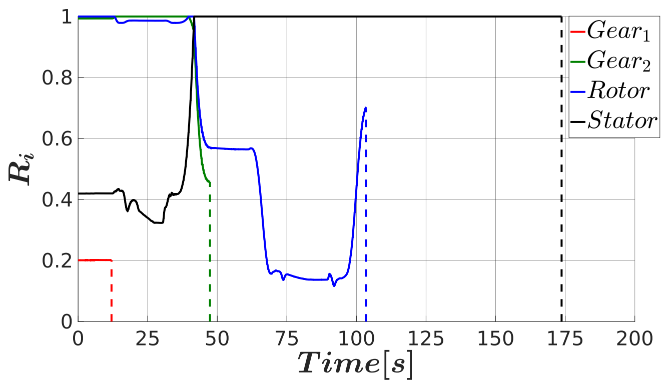
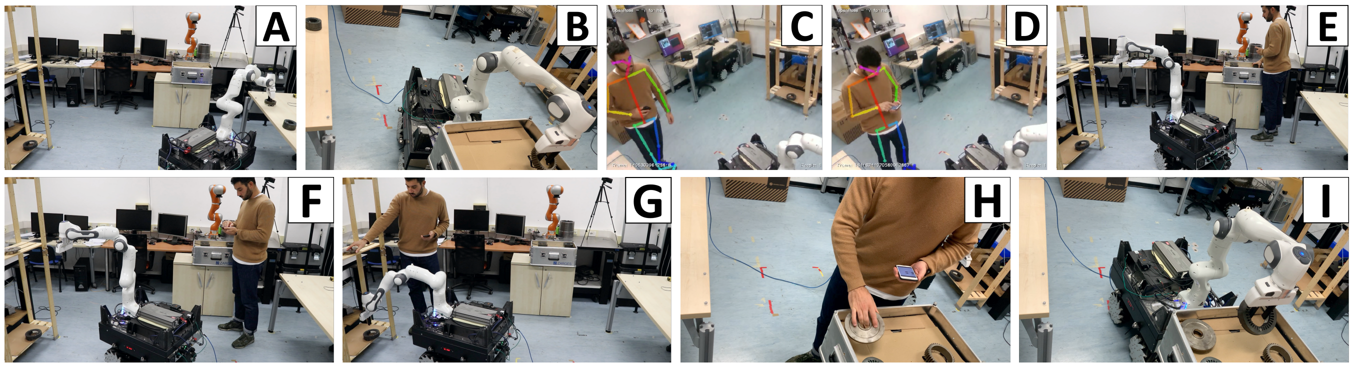


Fig. 11: Travel distance Costs of the 4 tasks of the experiment in the autonomous scenario. Weights: $\rho = 1$ and $\eta = \xi = \upsilon = 0$.

calculate the task-related distances and to estimate the height of the shoulder, waist and knee of the human, needed to compute the ergonomic index. The position of the object was considered known. To estimate the relative transformation between objects and MOCA we employed a motion capture system, OptiTrack. In this way, it was also possible to avoid large drifts on the odometry localization of the robot. Moreover, we developed and endowed the worker with a *web GUI* working on the browser of the cellphone. Thanks to the web GUI, the worker could inform the system about the intention to achieve a specific task, simply by pressing the task-related button (see Figure 9).

The BT of the experiments is illustrated in Figure 10. To prove that with the same BT it is possible to handle different situations, such as cooperation and autonomous task execution, we envision two different scenarios. In the first scenario, no workers are present in the work-cell and MOCA has to accomplish autonomously all the tasks. Then, we re-



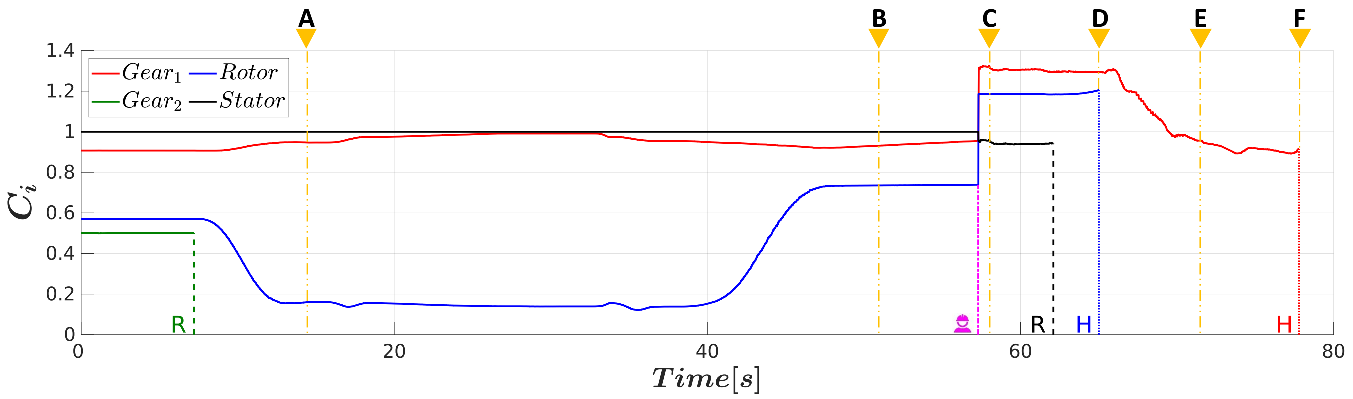
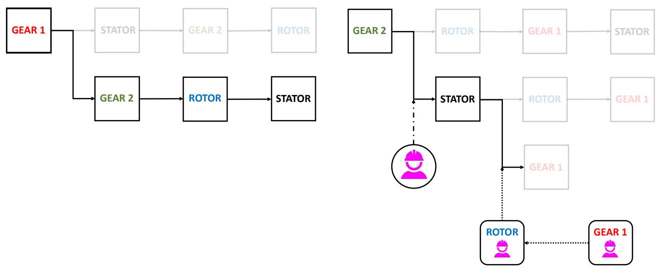


Fig. 12: Snapshots of the experiment. Since no workers are available, the BT plan allows MOCA to achieve autonomously the 4 tasks. MOCA grasps $gear_2$ (A), since it is the closest to the initial position, and immediately places it in the box (B). When the human enters in the scene (C), MOCA updates the weights of the cost function and his next task changes to the placement of the *stator*. The worker selects the *rotor* in the GUI (D), and completes its placement in the box (E). In the meanwhile, MOCA moves towards the *stator*. The worker selects the $gear_1$ while MOCA still moves towards the *stator* (F). At this point the worker could still select the *stator*, even if this action is less ergonomic. While MOCA is approaching the *stator*, the worker grasps the $gear_1$ (G) and places it in the box (H). Finally, MOCA grasps and places the *stator* in the box (I). Below: Mixed costs of the 4 tasks of the experiment in the cooperative scenario. Before the human enters the scene the weights are: $\rho = 1$ and $\eta = \xi = \upsilon = 0$. After the detection the weights are automatically set to: $\eta = 0.51$, $\rho = 0.49$, $\xi = 0.6$ and $\upsilon = 0$. The dash-dot line in magenta indicates the instant in which the human is detected by the camera. A Video of the experiment is available in the multimedia extension.

peated the same experiment, asking a human subject to enter the work-cell and place some of the objects, acknowledging the decision with the web GUI. A video of the cooperative experiment is available in the multimedia extension. In the experimental settings, we will not consider task duration, since the task does not require the repetitive execution of the same task.

Results: In the first experiment, since no workers are present, the cost weights are set as: $\rho = 1$ and $\eta = \xi = \upsilon = 0$. In this way, only the robot distance to each task is considered. The time evolution of the costs is shown in Figure 11. Since the cobot starts next to the shelves, it picks $gear_1$ as the first task. Then, while MOCA moves towards the box with the grasped item, he acknowledges that there exist another task that is closer to him than the planned one. For this reason, the planned sequence of tasks is updated accordingly.

In the second experiment, the robot starts to execute the task autonomously. In this situation the weights are exactly the same as in the previous case ($\rho = 1$ and $\eta = \xi = \upsilon = 0$), while we change initial robot position. After the first object ($gear_2$) is grasped, the human enters the work-cell. Thus, the costs consider also the human-related quantities, with new weights: $\eta = 0.51$, $\rho = 0.49$, $\xi = 0.6$ and $\upsilon = 0$. Since, in the beginning, the human is not in the workspace, $gear_2$ is the closest object to MOCA and, hence, the one with the least cost (Figure 12). Then, while the cobot is placing the $gear_2$ in the box, the human is detected (dash-dot line in magenta). Because of that, the costs instantaneously change and the plan, depicted in Figure 13b, is updated. At this moment, the task with least cost is the *stator* that is the farthest from the human, and, at the same time, the nearest to the robot and with the largest ergonomic risk factor for the worker. Then, the worker decides to place the *rotor* (its closest task). To acknowledge this decision, the worker selects the task in the web GUI (this specific moment is illustrated in Figure 9). Once the button is pressed, the decision is communicated to the BT and the plan is online updated leaving the selected task to the human. As can be noticed in Figure 13b, the *rotor* and the $gear_1$ tasks are accomplished by the human subsequently. We would like to highlight that the experiments were conducted using only one BT. This proves that the presented method allows encoding different robot behaviors, depending on the considered cost



(a) Robot plan computed in the autonomous scenario with the travel distance in Figure 11.

(b) Robot plan computed in the cooperative scenario with the mixed cost in Figure 12.

Fig. 13: Plans of the experimental task.

metrics and weights. Moreover, the human-centric planning strategy adapts robot behavior to the human actions and intentions, executing unpleasant tasks and the riskiest for human ergonomics.

V. Discussion and Conclusion

In this work, we proposed a novel human-aware task planner taking advantage of the Behavior Trees paradigm. The approach enables the robot to plan online the execution of tasks while obtaining different robot behaviors in relation to the user choice. The developed methods allow considering costs in the selection of the task to execute. Moreover, we presented three different metrics, suitable for manufacturing environments, to compute the cost values. Therefore, the robot can plan adapting to the dynamic changes of the environment and, especially, to human intentions, motion, decisions and availability. The same structure permits to consider different levels of engagement between robots and humans: coexistence, cooperation and even autonomous task execution. The explained results showed the high potential of the developed methods in improving robot reactivity and flexibility and, at the same time, considering the human motion, decisions and ergonomics. Future works will focus on extending the approach to multi-robot and multi-human teams and on merging robot task planning with an interactive task allocator.

References

[1] A. Ajoudani *et al.*, "Smart collaborative systems for enabling flexible and ergonomic work practices [industry activities]," *IEEE Robotics Automation Magazine*, vol. 27, no. 2, pp. 169–176, 2020.

[2] T. Asfour *et al.*, "Armar-6: A high-performance humanoid for human-robot collaboration in real-world scenarios," *IEEE Robotics & Automation Magazine*, vol. 26, no. 4, pp. 108–121, 2019.

[3] A. Kheddar *et al.*, "Humanoid robots in aircraft manufacturing: The airbus use cases," *IEEE Robotics Automation Magazine*, vol. 26, no. 4, pp. 30–45, 2019.

[4] A. Cherubini *et al.*, "A collaborative robot for the factory of the future: Bazar," *The International Journal of Advanced Manufacturing Technology*, vol. 105, no. 9, pp. 3643–3659, 2019.

[5] N. Kashiri *et al.*, "Centauro: A hybrid locomotion and high power resilient manipulation platform," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1595–1602, 2019.

[6] K. I. Alevizos, C. P. Bechlioulis, and K. J. Kyriakopoulos, "Physical human–robot cooperation based on robust motion intention estimation," *Robotica*, vol. 38, no. 10, pp. 1842–1866, 2020.

[7] E. Lamon, M. Leonori, W. Kim, and A. Ajoudani, "Towards an intelligent collaborative robotic system for mixed case palletizing," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9128–9134.

[8] P. Balatti, F. Fusaro, N. Villa, E. Lamon, and A. Ajoudani, "A collaborative robotic approach to autonomous pallet jack transportation and positioning," *IEEE Access*, vol. 8, pp. 142 191–142 204, 2020.

[9] I. El Makrini *et al.*, "Working with walt: How a cobot was developed and inserted on an auto assembly line," *IEEE Robotics & Automation Magazine*, vol. 25, no. 2, pp. 51–58, 2018.

[10] I. El Makrini, K. Merckaert, D. Lefeber, and B. Vanderborght, "Design of a collaborative architecture for human-robot assembly tasks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1624–1629.

[11] E. Lamon, F. Fusaro, P. Balatti, W. Kim, and A. Ajoudani, "A visuo-haptic guidance interface for the mobile collaborative robotic assistant (moca)," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, p. forthcoming.

[12] G. Buisan, G. Sarthou, and R. Alami, "Human aware task planning using verbal communication feasibility and costs," in *The 12th International Conference on Social Robotics (ICSR 2020)*, 2020.

[13] V. Montreuil, A. Clodic, M. Ransan, and R. Alami, "Planning human centered robot activities," in *2007 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2007, pp. 2618–2623.

[14] M. Cirillo, L. Karlsson, and A. Saffiotti, "Human-aware task planning for mobile robots," in *2009 International Conference on Advanced Robotics*. IEEE, 2009, pp. 1–7.

[15] S. Alili, M. Warnier, M. Ali, and R. Alami, "Planning and plan-execution for human-robot cooperative task achievement," in *19th international conference on automated planning and scheduling*, 2009.

[16] H. S. Koppula, A. Jain, and A. Saxena, "Anticipatory planning for human-robot teams," in *Experimental robotics*. Springer, 2016.

[17] M. Helmert, "A planning heuristic based on causal graph analysis." in *ICAPS*, vol. 16, 2004, pp. 161–170.

[18] A. A. Al-Moadhen, M. Packianather, R. Setchi, and R. Qiu, "Robot task planning in deterministic and probabilistic conditions using semantic knowledge base," *International Journal of Knowledge and Systems Science (IJKSS)*, vol. 7, no. 1, pp. 56–77, 2016.

[19] W.-Y. Loh, "Classification and regression trees," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 14–23, 2011.

[20] P. Allgeuer and S. Behnke, "Hierarchical and state-based architectures for robot behavior planning and control," *arXiv preprint arXiv:1809.11067*, 2018.

[21] M. Colledanchise and P. Ögren, "Behavior trees in robotics and ai: an introduction," *arXiv preprint arXiv:1709.00084*, 2017.

[22] C. Paxton, F. Jonathan, A. Hundt, B. Mutlu, and G. D. Hager, "Evaluating methods for end-user creation of robot task plans," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6086–6092.

[23] F. Rovida, B. Grossmann, and V. Krüger, "Extended behavior trees for quick definition of flexible robotic tasks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6793–6800.

[24] H. Zhou, H. Min, and Y. Lin, "An autonomous task algorithm based on behavior trees for robot," in *2019 2nd China Symposium on Cognitive Computing and Hybrid Intelligence (CCHI)*. IEEE, 2019, pp. 64–70.

[25] E. Giunchiglia, M. Colledanchise, L. Natale, and A. Tacchella, "Conditional behavior trees: Definition, executability, and applications," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, 2019, pp. 1899–1906.

[26] B. Merrill, "Building utility decisions into your existing behavior tree," *Game AI Pro 360: Guide to Architecture*, pp. 81–90, 2019.

[27] M. L. McShaffry, *Behavioral mathematics for game AI*. Cengage Learning, 2009.

[28] T. R. Waters, V. Putz-Anderson, A. Garg, and L. J. Fine, "Revised niosh equation for the design and evaluation of manual lifting tasks," *Ergonomics*, vol. 36, no. 7, pp. 749–776, 1993.

[29] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7291–7299.