Hierarchical Task Planning for Human-Robot Collaborative Assembly

Pegoraro, Giulia; Giacomuzzo, Giulio; Terreran, Matteo; Ghidoni, Stefano; Carli, Ruggero; Romeres, Diego

TR2025-134 September 16, 2025

Abstract

We propose E-UHTP, an enhanced task planner for collaborative assembly in communication-free human-robot environments. E-UHTP extends User-aware Hierarchical Task Planning (UHTP) by supporting joint actions, failure recovery, and online replanning. We introduce an automatic Hierarchical Task Network (HTN) generation method from annotated video demonstrations. Experimental simulations in multiple assembly scenarios demonstrate the improved performance, flexibility, and robustness of E-UHTP over baseline planners.

IFAC Joint Conference on Computers, Cognition, and Communication 2025

Hierarchical Task Planning for Human-Robot Collaborative Assembly

Giulia Pegoraro [*] **Giulio Giacomuzzo** [*] **Matteo Terreran** [*]
Stefano Ghidoni [*] **Ruggero Carli** [*] **Diego Romeres** [**]

[*] *Department of Information Engineering, University of Padova, Italy (e-mail: giacomuzzo@dei.unipd.it)*
[**] *Mitsubishi Electric Research Lab (MERL), Cambridge (MA), USA*

Abstract: We propose E-UHTP, an enhanced task planner for collaborative assembly in communication-free human-robot environments. E-UHTP extends User-aware Hierarchical Task Planning (UHTP) by supporting joint actions, failure recovery, and online replanning. We introduce an automatic Hierarchical Task Network (HTN) generation method from annotated video demonstrations. Experimental simulations in multiple assembly scenarios demonstrate the improved performance, flexibility, and robustness of E-UHTP over baseline planners.

Keywords: Human-Robot Collaboration, Task Planning, Hierarchical Task Networks (HTN), Collaborative Assembly, Task Representation, Failure Recovery, Joint Actions, Autonomous Systems.

1. INTRODUCTION

Technological innovation has profoundly reshaped the industrial landscape, integrating advanced automation and artificial intelligence into manufacturing processes. Among the most promising developments is Human-Robot Collaboration (HRC), a paradigm that combines the distinctive abilities of human workers, such as dexterity and decision-making, with the precision, strength, and repeatability of robots. Unlike traditional industrial robots, collaborative robots, or cobots, are designed to safely share workspaces with humans, enhancing workflow efficiency without the need for physical barriers.

A particularly significant area for HRC is collaborative assembly. Assembly processes typically involve long sequences of subtasks, alternating between heavy manipulations and delicate operations. These processes demand efficient coordination between humans and robots, continuous monitoring of actions, and dynamic task planning to manage the sequential dependencies and constraints. However, most existing frameworks rely on strict predefined plans (Fusaro et al., 2021; Darvish et al., 2018; Johannsmeier and Haddadin, 2016) or explicit communication between partners (Roncone et al., 2017; Shah et al., 2011; Milliez et al., 2016), which can reduce naturalness, flexibility, and overall collaboration efficiency. In real human teams, intentions are often inferred by observing actions rather than through constant verbal exchanges, and ideally, robots should exhibit similar adaptive capabilities.

This work introduces the *Extended User-aware Hierarchical Task Planner* (E-UHTP), a novel planning framework designed to support flexible, robust, and communication-free collaboration in realistic assembly environments. E-UHTP enables collaborative robots to plan and adapt in response to human behavior, while explicitly addressing two key limitations of existing approaches: the lack of support for joint human-robot actions, and the inability to manage task execution failures. By incorporating mechanisms for synchronized operations and autonomous failure handling, E-UHTP enhances both the expressiveness and resilience of collaborative planning.

E-UHTP builds upon the foundational *User-aware Hierarchical Task Planning* (UHTP) framework (Ramachandruni et al., 2023), which marked an important step toward natural human-robot collaboration by allowing the robot to observe the human partner and plan its actions without relying on predefined paths or explicit communication. UHTP relies on the representation of collaborative assembly tasks through Hierarchical Task Networks (HTNs) (Erol et al., 1994), which provide a structured and interpretable means to model sequential and parallel task dependencies. The robot continuously monitors the human's actions and autonomously selects its next step based on the evolving task state and constraints, promoting flexible interaction.

Nevertheless, UHTP assumes flawless execution and independent task execution, making it ill-suited for realistic settings that involve joint operations or unexpected disruptions. Moreover, the manual definition of HTNs can be labor-intensive and demands domain expertise, particularly for complex assemblies involving a large number of interdependent actions.

To overcome these limitations, E-UHTP contributes three main advancements. First, it includes an automated method for deriving HTNs from annotated video demonstrations, streamlining the creation of hierarchical task representations. Second, it extends the planner to support joint actions, enabling robots to participate in synchronized or complementary activities with their human

[1] Corresponding contact: giacomuzzo@dei.unipd.it

partners. Third, it incorporates a failure management mechanism that allows the robot to autonomously detect, respond to, and replan around errors or unforeseen events during task execution.

The proposed system retains the optimization-based, interpretable nature of UHTP while significantly enhancing its flexibility and robustness in dynamic collaborative settings. Simulated assembly experiments confirm that the extended framework improves task success rates, adaptability, and efficiency compared to baseline approaches. Together with random toy cases, we simulate the assembly of an IKEA chair as one of the use cases for the proposed algorithm, whose HTN is reported in Fig. 2.

2. PROBLEM STATEMENT AND BACKGROUND

In this Section, we first formalize the collaborative assembly problem we aim to solve, then we review HTNs, the representation formalism at the core of our method.

2.1 Problem Statement

Assembly tasks can be seen as a hierarchical set of actions, hereafter denoted as *primitive actions*, which must be executed in compliance with a set of ordering constraints. We assume two agents concur to complete the task: a human operator and a robot. Each agent can perform a subset of the primitive actions required to complete the assembly. We assume actions to be either individual or joint. Individual actions can be executed by a single agent, either the human or the robot. Joint actions must be executed by both agents together. We assume the human operator to be an uncontrolled agent, to which the robot should adapt.

Formally, we denote with $\mathcal{A}_h$ and $\mathcal{A}_r$ the finite sets of primitive actions that can be performed individually by the human and the robot, respectively. We enrich $\mathcal{A}_h$ and $\mathcal{A}_r$ with an *idle* action, to account for agent wait time. The set of joint actions, instead, is denoted as $\mathcal{A}_j$. While $\mathcal{A}_h$ and $\mathcal{A}_r$ are not mutually exclusive, namely there are individual actions that both agents can perform, actions in $\mathcal{A}_j$ cannot be contained in either $\mathcal{A}_h$ or $\mathcal{A}_r$. For later convenience, we define as *concurrent* the actions $a_c \in \mathcal{A}_h \cap \mathcal{A}_r$, namely the actions that can be performed both by the human and the robot.

We attribute to each action $a_h \in \mathcal{A}_h$, $a_r \in \mathcal{A}_r$ and $a_j \in \mathcal{A}_j$ a cost $c(a_h)$, $c(a_r)$, $c(a_j)$, respectively. Concurrent actions $a_c \in \mathcal{A}_h \cap \mathcal{A}_r$ has different costs if they are executed by the human or by the robot, which will be denoted as $c^h(a_c)$ and $c^r(a_c)$, respectively. For the sake of simplicity, in this work, we define the cost function only in terms of execution time.

The goal of the task planning algorithm is to find the robot plan that minimizes the overall task cost. Let π_h and π_r be the sequences of actions performed by the human and the robot. Note that each $a_h \in \mathcal{A}_h$, $a_r \in \mathcal{A}_r$ can only be present either in π_h or π_r, while each $a_j \in \mathcal{A}_j$ *must* be present in both π_h and π_r. The overall task cost is defined as

$$c_t = \sum_{a \in \pi_r} c(a) + \sum_{a \in \pi_h} c(a) \tag{1}$$

The challenge in the considered setup is that the human actions are not controllable by the system. For this reason, when planning, all the possible human choices should be in principle considered. To simplify the setup, we introduce some assumptions. For any a_c, we assume to know the probability of it to be performed by the human, and denote it with $p_h(a_c)$, such that $p_h(a_c) \in (0, 1)$. Accordingly, we define the probability of a_c to be performed by the robot as $p_r(a_c) = 1 - p_h(a_c)$. Note that such probabilities can be inferred based on prior knowledge, and can be eventually adapted online from run to run.

Finally, we assume a monitoring system to be available, which observes human actions and also provides information on actions' success or failure. Regarding handling of failures, we assume each action a to have a corresponding failure recovery action a^f, which *must* be executed after a failed.

2.2 Hierarchical Task Networks

Hierarchical Task Networks are a well-established formalism for representing procedural knowledge, particularly suited for planning problems. An HTN decomposes a complex assembly task into a hierarchy of simpler subtasks. The network typically consists of non-primitive tasks, which need further decomposition, and primitive tasks, which correspond to executable actions. Subtasks can be ordered, imposing sequential constraints, or unordered, allowing for parallel execution or flexible ordering.

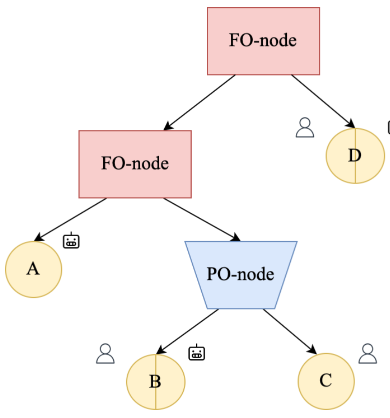

Fig. 1. Example HTN. Nodes represent tasks (internal nodes) or primitive actions (leaves). Ordering constraints are expressed through Partially Oriented (PO) or Fully Oriented (FO) nodes.

HTNs can be represented as trees. A pictorial example of HTN is reported in Figure 1. The root represents the overall assembly goal. Internal nodes can be non-primitive tasks requiring decomposition or control nodes specifying execution constraints. We utilize two primary control nodes: Fully Ordered (FO) nodes, whose children must be executed sequentially from left to right, and Partially Ordered (PO) nodes, whose children can be executed in any order or potentially in parallel, provided their individual preconditions are met. Primitive actions form the leaves of the tree. An assembly is considered completed when all the primitive actions have been completed.

3. EXTENDED USER-AWARE HIERARCHICAL TASK PLANNER

In this Section, we present our contributions, the E-UHTP framework. In particular, in Section 3.1 we detail the E-UHTP task planner we propose. Then, in Section 3.2 we describe an automatic HTN generation algorithm, which we use to derive HTNs from assembly videos.

3.1 E-UHTP Implementation

As anticipated, the challenge when planning in the considered setup is the human not being controllable. The challenge is further exacerbated by the occurrence of failures, which cannot be predicted but need to be taken into account. The core idea of E-UHTP is to plan robot actions by considering the current human's action as well as all their possible future choices, while reacting in case of failures' occurrence.

The workflow of the algorithm is reported in Algorithm 1.

Algorithm 1. E-UHTP program flow
```
 1: function E-UHTP($\mathcal{H}$)
 2:     $\mathcal{H}^{EUHTP} \leftarrow$ EXTEND($\mathcal{H}$)
 3:     $a_h \leftarrow idle$
 4:     $a_r \leftarrow idle$
 5:     while $\mathcal{H}^{EUHTP}$ is not empty do
 6:         $a'_h \leftarrow$ ACTIVITYRECOGNITION()
 7:         if $a'_h \neq a_h$ then
 8:             if $a_h$ succeded then
 9:                 $\mathcal{H}^{EUHTP}$.REMOVE($a_h$)
10:                 PRUNEBRANCHES($\mathcal{H}^{EUHTP}$, $a'_h$)
11:             else
12:                 $\mathcal{H}^{EUHTP}$.HANDLEFAILURE($a_h$)
13:             end if
14:             $a_h \leftarrow a'_h$
15:         end if
16:         if robot is idle then
17:             if $a_r$ succeded then
18:                 $\mathcal{H}^{EUHTP}$.REMOVE($a_r$)
19:                 PRUNEBRANCHES($\mathcal{H}^{EUHTP}$)
20:             else
21:                 $\mathcal{H}^{EUHTP}$.HANDLEFAILURE($a_r$)
22:             end if
23:             if $a_h \in \mathcal{A}_j$ then
24:                 $a_r \leftarrow a_h$
25:             else
26:                 $a_r \leftarrow$ SELECTMINACTION($\mathcal{H}^{EUHTP}$)
27:             end if
28:             EXECUTE($a_r$)
29:         end if
30:     end while
31: end function
```

HTN Extension Similarly to what proposes the UHTP framework (Ramachandruni et al., 2023), the standard HTN structure described in Section 2 is first extended by assigning potential actors (human or robot) to primitive actions, and by adding costs to each node, which produces a novel extended HTN which we call $\mathcal{H}^{EUHTP}$. In detail, if an action is concurrent, the corresponding primitive action in the HTN is replaced by a novel type of node called Decision (D) node (see Figure 2), with agent-specific primitive action nodes as children. Moreover, in order to address joint actions, in the E-UHTP framework we introduce an additional agent, hereafter denoted as *joint*.

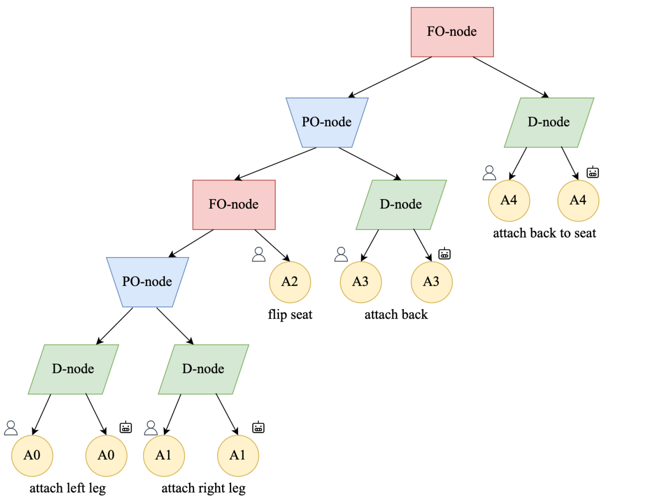

Fig. 2. Example HTN representing a chair assembly. Nodes represent tasks (internal nodes) or primitive actions (leaves). Ordering constraints are expressed through Partially Oriented (PO) or Fully Oriented (FO) nodes.

Concerning cost computation, note that the overall task cost in (1) requires knowing the entire sequence of action taken by both the agents, which is not possible in the considered setup. We can only estimate the cost based on the available information. To this aim, we first assign a cost to each node in the HTN. Costs are computed recursively bottom-up, from leaves to root. For each node n we compute three costs: agent-specific costs $c_h(n)$ and $c_r(n)$, that encodes the cost for each single agent executing the node, and a total cost $c_{tot}(n)$, which combines agent-specific costs and is exploited to select the robot action.

For Primitive nodes, the cost is the expected time duration for the corresponding agent, and zero for the other. The total cost corresponds to the non-zero agent-specific cost. The cost of a D node are the probability-weighted sum of its children's costs. Assume n_d to be the decision node generated by the concurrent action a_c. Then,

$$c_{tot}(n_d) = p_h(a_c)c^h(a_c) + p_r(a_c)c^r(a_c)$$

while

$$c_h(n_d) = p_h(a_c)c^h(a_c) \text{ and } c_r(n_d) = p_r(a_c)c^r(a_c).$$

The costs of FO node, instead, is the sum of its children's costs, that is

$$c_{tot}(n_f) = \sum_n c_{tot}(n),$$

while

$$c_h(n_f) = \sum_n c_h(n) \text{ and } c_r(n_f) = \sum_n c_r(n),$$

where n_f represents a FO node, and the set $\{n | n \in nf\}$ is the set of children of n_f. Finally, the costs of a PO node n_p requires considering the possibility of actions to be executed contemporary. First, we compute the agent-specific costs as for the FO nodes, namely as the sum of the agent-specific costs of the node's children. Regarding the total cost, we compute the lower bound of $c_{tot}(n_p)$ as

$$\text{lb}(n_p) = \max(\{c_{tot}(n) | n \in n_p\}, c_h(n_p), c_r(n_p))$$

The intuition behind this is that performing actions contemporary should take at least the time required by the slowest action, and cannot be faster than single-agent execution times.

$$c_r = c_{tot} = c$$

Then, we compute the upper bound as if the children of n_p were executed sequentially, that is

$$\text{ub}(n_p) = \sum_n c_{tot}(n).$$

Finally, the total cost is estimated as the average

$$c_{tot}(n_p) = \frac{\text{lb}(n_p) + \text{ub}(n_p)}{2}.$$

Robot Actions Planning When the human successfully completes an action (see algorithm 1, line 7-14), the extended HTN is updated by removing branches corresponding to the completed action and any now invalid alternative. Specifically, all the primitive nodes containing the completed action a_h are removed from $\mathcal{H}^{EUHTP}$, together with any decision branch that is inconsistent with a'_h, using the `pruneBranches` function described (Ramachandruni et al., 2023).

Similarly, when the robot becomes idle, if its last action has been successfully completed, the $\mathcal{H}^{EUHTP}$ task model is updated. Then, if the human is not triggering a joint action, the robot's next action is selected, through `SelectMinAction` (see algorithm 2), by evaluating all the currently available robot actions. Valid actions include any action $a_r \in \mathcal{A}_r$, that is reachable by following the *first* child of FO nodes and *any* child of PO or D nodes. Executing each available action a^i_r would lead to a different HTN $\mathcal{H}^i$, which can be obtained from $\mathcal{H}^{EUHTP}$ by pruning the decision branches inconsistent with the execution of a^i_r. For each $\mathcal{H}^i$, the corresponding cost can be computed using the methodology described above. Then, comparing the total cost of the root nodes $c_{tot}(r^i)$, with r^i being the root of $\mathcal{H}^i$, E-UHTP selects the next robot action as

$$a_r = \min_{a^i_r} c_{tot}(r^i).$$

Algorithm 2. Pseudocode for the SelecMinAction function.

```
1:  function SELECTMINACTION($\mathcal{H}^{EUHTP}$)
2:      ar ← GETVALIDROBOTACTIONS(root($\mathcal{H}^{EUHTP}$))
3:      $\mathcal{H}'$ ← []
4:      for all $a_r$ in a$_r$ do
5:          $\mathcal{H}_{tmp}$ ← COPY($\mathcal{H}^{EUHTP}$)
6:          PRUNEBRANCHES($\mathcal{H}_{tmp}, a_r$)
7:          AGGREGATECOSTS($\mathcal{H}_{tmp}$)
8:          $\mathcal{H}'$.add($\mathcal{H}_{tmp}$)
9:      end for
10:     $i_{min}$ ← $\arg\min_i c_{tot}(\text{root}(\mathcal{H}'[i]))$
11:     return a$_r[i_{min}]$
12: end function
```

The innovation of the method described above is represented by the combination of the action optimization and the HTN tree pruning. Optimization over entire trees, indeed, could become particularly expensive, in particular for long-lasting and complex assemblies. Removing all the impossible sequences each time an action is selected represents a simple yet effective solution to maintain the complexity tractable. We stress the fact that, so far, the functioning of the E-UHTP framework closely resembles that of (Ramachandruni et al., 2023). In the next paragraphs, we describe how we integrated joint actions and failure handling.

Failure Recovery Failure Recovery is implemented to handle unexpected action failures. As anticipated, we assume failures to be detected by the monitoring system. Furthermore, for the sake of simplicity, we assume that each primitive action has a corresponding recovery action to be executed in case of failure. The problem of making the robot autonomous in deciding how to handle failures is an attractive yet very complex aspect, which is left as possible future work. To handle failures, we introduce a novel node type called Recovery (R) node. Upon failure detection, the failed action node is temporarily removed from the HTN, and an R-node is inserted into the HTN as a child of the failed action's parent. This R-node has its own associated cost and duration and may represent tasks like picking up a dropped part or correcting a misalignment. The completion of the recovery task triggers the re-insertion of the original failed action node into the HTN, allowing it to be attempted again when its requirements are met. The framework ensures that subsequent dependent actions cannot proceed until the recovery and the original action are successfully completed. Note that the mechanism described above and implemetned through the `HandleFailure` function in algorithm 1, makes the HTN vary dynamically, according to the outcomes of the monitoring system, which introduces more flexibility and robustness to the framework. This mechanism can be easily extended to handle any kind of unpredictable event, such as human changes of mind or agents' inabilities to complete a task.

Joint Actions Joint Actions are incorporated to account for tasks requiring simultaneous effort. We assume that only the human agent can initiate a joint action. When the human initiates a joint action (see algorithm 1, lines 23-24), and robot is idle, it immediately starts executing the same joint action, synchronized with the human. If the robot is busy with another task, it completes the ongoing task first and then joins the human in to perform the joint action. Failures during joint actions are handled by the standard recovery mechanism, potentially requiring both agents to perform the recovery together.

3.2 HTN Construction from Annotated Videos

Automating HTN generation reduces manual effort and can capture realistic task structures. Our approach takes as input a dataset of annotated videos representing successful assembly executions. Each video is represented as a sequence of tuples, where each tuple contains the actions being performed simultaneously by the human and the robot, or 'idle' if an agent is inactive. Each action is described by its name, its duration in time units, and the agent performing it. Joint actions are represented by identical action names for both agents in a tuple.

In the following we will consider a chair assembly example. The chair assembly is constituted by 5 primitive acions:

A0: Attach left leg;
A1: Attach right leg;
A2: flip seat;
A3: Attach back;
A4: Attach back to seat;

A possible HTN representing the task is reported in Fig. 2. An example of tuple representing a part of an annotated video is reported below.

```
  (["attach left leg",[3],["human"]],
  ["attach right leg",[2],["robot"]])
```

The algorithm proceeds in two main stages. First, it extracts action requirements, that is, for each primitive action observed in the dataset, the algorithm determines the set of other actions that must be completed before it can start. To this aim, for each video sequence, a list of actions completed so far is maintained. When an action occurs, the set of already completed actions represents the potential set of preconditions. By analyzing multiple videos showing different valid execution paths, the algorithm identifies the necessary preconditions for each action as the intersection of the sets of completed actions observed just before the action starts. An example of the results returned by this process is reported below for one of the actions of the chair example. The list reports the action name, the nominal durations extracted from the videos, the agents able to perform the action, and the requirements.

```
["attach back to seat",
[5,7],
["human","robot"],
["attach right leg",
  "attach left leg",
  "attach back",
  "flip seat"]]-
```

Second, the algorithm constructs the HTN tree structure based on the requirements. Note that, while the requirements set derived from an HTN is unique, multiple valid HTN structures might be possible from a given set of requirements, depending on how subtasks are grouped. Our approach constructs one of the possible HTNs. The construction proceeds bottom-up from the leaves (primitive actions). Actions with no requirements can be placed first. Actions whose requirements are fully met by the actions already placed can then be added. If an action has been seen to be performed by both a human and a robot, a decision node is created. When multiple actions become available simultaneously (i.e., their requirements are met by the same set of completed actions), they are grouped under a PO node if their requirements do not impose a sequence between them, signifying potential parallelism. If the requirements dictate a specific order, instead, they are placed sequentially under a FO node. This process recursively builds the tree until the root node, representing the final assembly goal, is reached.

4. EXPERIMENTAL EVALUATION

We conducted simulation experiments to validate the proposed methods. To simulate the assembly tasks, we modeled the human as a random agent. The duration of the actions is modeled as a Gaussian distribution with nominal values of the action duration as mean and a standard deviation set to 5 % of the nominal duration.

4.1 Failure Recovery Robustness Evaluation

To assess the robustness introduced by the failure recovery mechanism, we simulated task executions under varying conditions of uncertainty. We considered the chair assembly task, systematically increasing the probability of failure for each primitive action from 10% up to 50%. For each probability level, 100 runs were performed, recording the total task completion time. Table 1 reports the completion times. As expected, the mean and variance of the completion time increase with higher failure probabilities, reflecting the time spent on recovery actions and re-attempts. Despite the increased execution time, the extended UHTP framework successfully completed the assembly task in all simulation runs across all tested failure rates and task complexities. This demonstrates that the implemented recovery process ensures task completion, enhancing the framework's robustness.

Table 1. E-UHTP Completion Time at Different Failure Probabilities on the Chair Example

Failure Prob[%]	10%	20%	30%	40%	50%
Completion Time [s]	27	28	34	40	46

4.2 Planning Policy Comparison

We evaluated the efficiency of the E-UHTP framework against two baseline policies: a Greedy policy, where the robot selects the available action with the lowest immediate cost (duration), and a Random policy, where the robot randomly selects among its currently available actions.

We ran 1000 simulations for each policy on three different tasks: the chair assembly and two randomly generated tasks with 16 and 32 actions, respectively, featuring varying requirement complexities. Within the randomly generated tasks, we also randomly included joint actions. Table 2 presents the mean task completion times. Across all tested scenarios, the E-UHTP policy consistently achieved the lowest average completion time. While the Greedy policy performed better than Random, it was often suboptimal, as minimizing immediate cost does not guarantee minimizing total assembly time. The E-UHTP policy's ability to consider the aggregated cost of the remaining task tree allowed for making more globally efficient decisions, effectively coordinating with the unpredictable human agent to minimize overall duration. The standard deviation was also generally comparable or lower for E-UHTP, suggesting consistent robustness to randomization.

5. CONCLUSIONS

This paper addressed key challenges in task representation and planning for human-robot collaborative assembly

Table 2. Task Completion Time Comparison

Policy	16 Actions	32 Actions	Chair
Greedy	178.75 ± 11.10	340.71 ± 14.21	22.52 ± 0.65
Random	187.38 ± 12.33	344.26 ± 14.60	22.95 ± 0.42
E-UHTP	**172.26 ± 7.08**	**331.10 ± 11.32**	**22.16 ± 0.26**

within a communication-free setting. We introduced two main contributions to enhance the flexibility and robustness of such systems. First, we developed and validated an algorithm capable of automatically constructing Hierarchical Task Network representations from annotated video demonstrations. Second, we presented an extension of the state-of-the-art UHTP framework, named E-UHTP. E-UHTP incorporates mechanisms for handling joint actions and for recovering from action failures. The framework maintains the core UHTP principle of optimizing robot actions based on aggregated costs while treating the human as an unpredictable agent, thus avoiding restrictive human modeling or explicit communication requirements.

ACKNOWLEDGEMENTS

Giulio Giacomuzzo was supported by PNRR research activities of the consortium iNEST (Interconnected North-Est Innovation Ecosystem) funded by the European Union Next GenerationEU (Piano Nazionale di Ripresa e Resilienza (PNRR) – Missione 4 Componente 2, Investimento 1.5 – D.D. 1058 23/06/2022, ECS_00000043). This manuscript reflects only the Author's views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

REFERENCES

Darvish, K., Wanderlingh, F., Bruno, B., Simetti, E., Mastrogiovanni, F., and Casalino, G. (2018). Flexible human–robot cooperation models for assisted shop-floor tasks. In *Mechatronics*, volume 51, 97–114. Elsevier.

Erol, K., Hendler, J.A., and Nau, D.S. (1994). UMCP: A sound and complete procedure for hierarchical task-network planning. In *Proceedings of the 2nd International Conference on AI Planning Systems (AIPS'94)*, 249–254.

Fusaro, F., Lamon, E., De Momi, E., and Ajoudani, A. (2021). A human-aware method to plan complex cooperative and autonomous tasks using behavior trees. In *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, 522–529. IEEE.

Johannsmeier, L. and Haddadin, S. (2016). A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes. *IEEE Robotics and Automation Letters*, 2(1), 41–48.

Milliez, G., Lallement, R., Fiore, M., and Alami, R. (2016). Using human knowledge awareness to adapt collaborative plan generation, explanation and monitoring. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 43–50. IEEE.

Ramachandruni, K., Kent, C., and Chernova, S. (2023). UHTP: A user-aware hierarchical task planning framework for communication-free, mutually-adaptive human-robot collaboration. *ACM Transactions on Human-Robot Interaction*, 12(4), Article 43, 1–28.

Roncone, A., Mangin, O., and Scassellati, B. (2017). Transparent role assignment and task allocation in human robot collaboration. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 1014–1021. IEEE.

Shah, J., Wiken, J., Williams, B., and Breazeal, C. (2011). Improved human-robot team performance using chaski, a human-inspired plan execution system. In *Proceedings of the 6th international conference on Human-robot interaction (HRI '11)*, 29–36. ACM.