Human-Aware Robot Task Planning Based on a Hierarchical Task Model

Yujiao Cheng, Liting Sun, and Masayoshi Tomizuka, *Member, IEEE*

***Abstract*—When robots work with humans for collaborative task, they need to plan their actions while taking humans' actions into account. However, due to the complexity of the tasks and stochastic nature of human collaborators, it is quite challenging for the robot to efficiently collaborate with the humans. To address this challenge, in this letter, we first propose an algorithm to automatically construct a hierarchical task model from single-agent demonstrations. The hierarchical task model explicitly captures the sequential and parallel relationships of the task at all levels of abstraction. We then propose an optimization-based planner, which exploits the parallel relationships and prioritizes actions that are parallel to the humans' actions. In such a way, potential spatial interfaces can be avoided, task completion time can be reduced, and human's satisfaction level can be improved. We conducted simulations of a robot arm collaborating with a human for several collaborative tasks. The comparison results with several baselines proved that our proposed planner is better in terms of efficiency, safety and human satisfaction.**

***Index Terms*—Industrial robots, human-centered robotics, assembly, task planning.**

I. Introduction

ROBOTIC systems are increasingly integrated into human workspace for collaborative tasks. In the assembly lines, for example, there is a strong economic motivation to enable robots and humans to cooperatively conduct work [1], [2]. Such integration raises two challenges for robotic systems: 1) perceiving humans' actions and predicting their intended tasks such as in [3], [4], and 2) planning robot's tasks while taking humans' actions into account, also known as human-aware task planning. So far, there have been many research efforts on human-aware task planning. Some works such as [5], [6], [7] solve this problem by empowering a human worker or a centralized planner to be a supervisor. The existence of supervisors makes the system less flexible and makes the human more stressed. Other works do not include supervisors. They build a robot planner which takes as input the human's actions, the human intention, environment states, task models, and etc., and enables the robot to collaborate as a teammate. The mechanism usually consists of two steps: offline obtaining task knowledge and online generating plans. In the literature, task knowledge can be acquired via engineering by experts [8]–[12] or learning from demonstration [13]. Online plans can be generated by techniques such as search algorithms [8], [9], [11], [12], optimization of the collaborative cost [10], or a Q-learning method [13]. In this letter, we adopt this two-step mechanism, where task knowledge is extracted from human demonstration offline, and robot actions are planned by optimization online.

Task knowledge is often described by graphical task models. Depending on the structure of the models, they can be categorized as flat models, such as a plan network [9], or hierarchical models, such as and/or graphs [14], [15]. Hierarchical models have shown superiority over flat models for human-robot collaboration, because levels of abstraction in hierarchical models are close to human intuitions, which can help predict actions of human and plan predicable actions of the robot. To our advantage, we propose a sequential/parallel task model which is a hierarchical task model. This model inherits the feature from the model in [16] that it explicitly models parallel relationships of subtasks and actions. To learn the hierarchical task models, Hayes and Scassellati in [17] proposed a conjugate task graph and an aggregation algorithm for identification of underlying structure of a task. Nevertheless, the introduction of conjugate task graph shrinks the task space, which makes it less applicable to some use cases. We propose to extract the sequential/parallel task model using the idea of aggregation but without introducing conjugate graphs.

Optimization-based planners formulate the planning problem as finding a robot action from all feasible actions that minimizes the collaborative costs including factors such as completion time [18], human fatigue [19], spatial interfaces [18], and etc.. The more factors considered, the more complex the cost function becomes, and it is nontrivial to decide on the weights for each factor. With poor modeling, the planning results can be far below expectations. We propose an optimization-based planner, the objective of which is to minimize the task completion time. Different from other methods, we give priorities to actions that are parallel to the human's. The perspective is that it is worth sacrificing some completion time for conducting parallel actions, because this has potential benefits in reducing spatial interfaces and improving human's satisfaction, which are not explicitly included in the objective function.

The key contributions of this letter are:

- We propose the sequential/parallel task model and a method to automatically extract the model from human demonstrations.

Manuscript received October 15, 2020; accepted January 26, 2021. Date of publication February 2, 2021; date of current version February 17, 2021. This letter was recommended for publication by Associate Editor C.-B. Yan and Editor J. Yi upon evaluation of the reviewers' comments. *(Corresponding author: Yujiao Cheng.)*

The authors are with the School of Engineering, Mechanical Engineering Department, University of California, Berkeley 94720, California USA (e-mail: yujiaocheng@berkeley.edu; litingsun@berkeley.edu; tomizuka@me.berkeley.edu).

Digital Object Identifier 10.1109/LRA.2021.3056370

2377-3766 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Univ of Calif Berkeley. Downloaded on June 19,2021 at 12:24:28 UTC from IEEE Xplore. Restrictions apply.

TABLE I
TABLE OF NOTATION

m	$\in N$	Number of actions in a demonstration.
a	$\in$ Action space	Action.
ξ	$[a_1, a_2, ..., a_m]$	Action sequence.
n	$\in N$	Number of action sequences.
Ξ	$\{\xi_1, \xi_2, ..., \xi_n\}$	Demonstration set.
s	$\in \{0,1\}^m$	Execution indicator.
req$(\cdot)$	$\in \{0,1\}^m$	Requisite execution indicator.
prod$(\cdot)$	$\in \{0,1\}^m$	Resulting execution indicator.
C_r	$\in$ Action space	Robot capability.
C_h	$\in$ Action space	Human capability.
k	$\in N$	Planning horizon.
x_r	$\in \{0,1\}^k$	Robot task assignment vector.
x_h	$\in \{0,1\}^k$	Human task assignment vector.

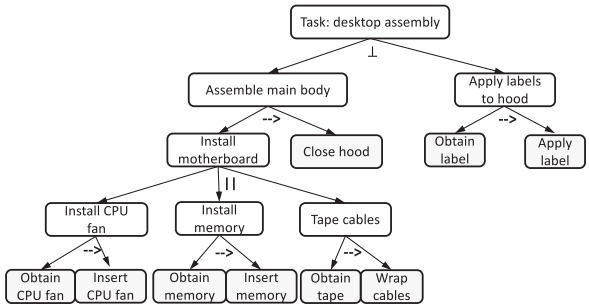

Fig. 1. The sequential/parallel task model for a desktop assembly task.

- We propose a robot planner that explicitly uses the parallel information in the sequential-parallel task model.
- We provide simulations to compare our planner with baseline planners, and show that our planner is better in terms of efficiency, safety and human satisfaction.
- We use the planner on our robot to work with a human for the desktop assembly task.

The main notation used in this letter is summarized in Table I. The rest of the letter is organized as follows. Section II introduces the sequential/parallel task model. Section III-A proposes the algorithm for automatically constructing sequential/parallel task model from human demonstrations, and Section III-B proposes a planner that takes advantage of the parallel information in our task model. In Section IV, simulations and experiments are included. Finally, Section V concludes the letter.

II. A HIERARCHICAL TASK MODEL

Hierarchical task models surpass the flat models in the fields of human-robot collaboration in three ways. First, hierarchical task models provide more intuitive abstractions of the task for the robots, which can help improve the human-robot communication about the intermediate goals of the task [20], [21]. Second, since humans follow the hierarchical abstractions when accomplishing a task, hierarchical task models can help predict the human's actions [22], [23]. Finally, it has been found that such hierarchical abstractions can be learned remarkably fast from relatively little data compared with what is needed for learning at lower levels due to overhypotheses [24], [25], [26]. Thus, in this work, we adopt a hierarchical task model named a sequential/parallel task model. Fig. 1 shows an example of the sequential/parallel task model for the desktop assembly task. The root node represents the *task*, and all the other nodes represents *subtasks*. Leaf nodes are *atomic subtask*, also known as *actions*. Nodes can be categorized to the following three types according to the relationship among their child nodes.

- **Sequential nodes**: their child nodes must be executed in the order from left to right, which is denoted by the operator "$\rightarrow$". For example, the subtask "Install CPU fan" is a sequential node, and its child "obtain CPU fan" must be conducted before "Insert CPU fan".
- **Parallel nodes**: their child nodes can be executed in parallel, which is denoted by "$\|$". For example, the subtask "Install motherboard" is a parallel node, its children "install CPU fan," "install memory" and "Tape cables" can be executed simultaneously.
- **Independent nodes**: their child nodes can be executed in any orders, which is denoted by "$\perp$". Parallel nodes are special case of independent nodes. For example, root node is an independent node but not a parallel node. Its child nodes "Applying labels to hood" and "Assemble main body" have no fixed order, but they cannot be executed in parallel if "close hood" is in progress.

Following the description of [3], [27], actions can be defined as $a = [\{motion, object\}, attribute]$, where *motion* indicates types of the movement and *object* indicates the object of interaction. The pair of *motion* (e.g. inserting) and *object* (e.g. CPU fan) is sufficient to distinguish different actions. In addition, the *attribute* contains information such as completion time, energy consumption, etc., which are useful in the planning process. For assembly tasks, the relationships of actions also take three forms.

- Two actions a and b are **independent** (written as $a \perp b$ or $b \perp a$) if the occurrence of one action does not rely on the occurrence of the other.
- Action a is **dependent** on the action b (written as $a \not\perp b$) if a can not occur in the absence of the occurrence of action b.
- Actions a and b are **parallel** (written as ab) if a and b are independent and they do not share objects of interaction.

An action a is dependent on the action b implies that b produces some consequences that a requires in order to be executed. We use the *execution indicator* to represent this connection, each component representing an action being executed or not. There are totally 9 actions in the example, and thus the indicator is $s \in \{0,1\}^9$. Note that for assembly tasks, the effects of several actions are addictive, thus indicator vectors can be used as procedural states of the assembly. Based on this, we define two types of indicator vectors, req $(\cdot)$ to be *requisite execution indicator* and prod$(\cdot)$ to be the *resulting execution indicator* of an action or an action sequence. In this sense, action a is dependent on action b ($a \not\perp b$) is equivalent to req$(a) \wedge$ prod(b) = prod (b), where $\wedge$ is a bit-wise AND operator. In addition, that action d is dependent on action sequence bcd ($d \not\perp abc$) is equivalent to req$(d) \wedge$ prod$(abc) =$ prod (abc). As the effects of actions are *addictive*, prod$(abc) =$ prod$(a) +$ prod$(b) +$ prod(c), hence req$(d) \wedge$ prod$(abc) =$ prod (abc) is equivalent to req$(d) \wedge$ prod$(a) =$ prod (a), req$(d) \wedge$ prod$(b) =$ prod (b) and req$(d) \wedge$

Authorized licensed use limited to: Univ of Calif Berkeley. Downloaded on June 19,2021 at 12:24:28 UTC from IEEE Xplore. Restrictions apply.

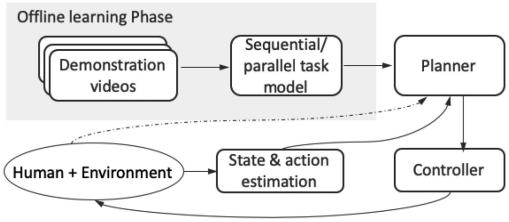

Fig. 2. System overview.

prod(c) = prod (c). Besides, for actions with independent relationships, action a is independent of action b ($a \perp b$) if and only if req$(a)\wedge$ prod$(b) = \mathbf{0}$ and req $(b)\wedge$ prod$(a) = \mathbf{0}$. Action d and action sequence abc is independent if and only if req $(d)\wedge$ prod$(abc) = \mathbf{0}$ and req $(abc)\wedge$ prod$(d) = \mathbf{0}$. Similarly, by the addition characteristic of the effect, req $(d)\wedge$ prod$(abc) = \mathbf{0}$ is equivalent to req $(d)\wedge$ prod$(a) = \mathbf{0}$, req $(d)\wedge$ prod$(b) = \mathbf{0}$ and req $(c)\wedge$ prod$(abc) = \mathbf{0}$.

III. Approach

Our goal is to develop a human-aware robot task planner for industrial assembly scenarios. There are two aspects to be addressed: (i) learning the knowledge of assembly plans from human demonstrations, and (ii) designing a human-aware robot task planner using the task knowledge. Fig. 2 shows the proposed robotic system. We first learn a sequential/parallel task model from human demonstrations at the offline learning phase. This model represents the plans of a task with levels of abstraction, which provides information about the parallel relationships that could not be directly seen in a collection of action sequences. At the online execution phase, the planner takes as input the task model and human actions, then outputs the robot action command to the controller to execute it. Our planner leverages the parallel information in the sequential/parallel task model to guide the robot towards performing actions parallel to the human's action, while optimizing over task efficiency.

A. Automatically Constructing Sequential/Parallel Task Models From Human Demonstrations

As the fast development in computer vision techniques, action recognition for videos can be accurate and fast using the state-of-the-art methods such as [28], [29], [30]. Applying such techniques to make annotations to each human demonstration video, a set of action sequences $\Xi = \{\xi_1, \xi_2, \ldots, \xi_n\}$ can be obtained, where n is the number of different plans demonstrated and $\xi_i = [a_1^i, a_2^i, a_3^i, \ldots, a_{m^i}^i]$ is the i-th action sequence with horizon m^i. For industry assembly tasks, the number of steps for any possible plan is often the same, thus we omit the superscript of m^i for simplicity. The set Ξ is complete if and only if it includes all the possible plans.

To construct the sequential/parallel task model from Ξ, the key is to discover the independent relationships and sequential relationships at all levels of abstraction. Once the independent relationships are identified, recognizing parallel relationships is trivial by checking objects of interaction. In the following, we first discuss how to find a sequential relationship among actions, and second we explain how to discover an independent relationship, and then we introduce a bottom-up algorithm which iteratively aggregates actions and forms a sequential/parallel task model. Finally we discuss whether there is a requirement on the completeness of the demonstration set and then we display time complexity.

1) Sequential Relationships: To identify the actions with sequential relationships is to find the longest common subsequence (LCS). The longest subsequence is common to all the given sequences $\xi_i \in \Xi, i \in \{1, 2, \ldots, n\}$, provided that the elements of the subsequence are required to occupy consecutive positions within the original sequences. By saying so, we impose an assumption that the sequential actions are not interrupted by parallel actions in human's demonstration. For example, "Apply labels to hood" cannot take place in between "Obtain CPU fan" and "Insert CPU fan". This is a weak assumption because reasonable humans follow this way when proceeding with tasks.

2) Independent Relationships: Identifying the independent relationships is based on the following theorem.

Theorem 1: When Ξ is complete, for a subsequence η of $\xi \in \Xi$, and suppose $\bar{\eta}$ is a reversed η, the following two statements are equivalent:

a) There exists a $\gamma \in \Xi$ such that $\bar{\eta}$ is a subseqence of γ.

b) Each pair of actions in η are independent.

Proof: (a) "$\Rightarrow$" (b)

Suppose subsequence $\eta = [b_1, b_2, \ldots, b_m], m \geq 2$, then $\bar{\eta} = [b_m, b_{m-1}, \ldots, b_1]$. From η, we have

$$req(b_1) \wedge prod(b_2 b_3 \ldots b_m) = \mathbf{0}$$
$$req(b_2) \wedge prod(b_3 b_4 \ldots b_m) = \mathbf{0}$$
$$\ldots$$
$$req(b_{m-1}) \wedge prod(b_m) = \mathbf{0},$$

and from $\bar{\eta}$, we have

$$req(b_m) \wedge prod(b_{m-1} b_{m-2} \ldots b_1) = \mathbf{0}$$
$$req(b_{m-1}) \wedge prod(b_{m-2} b_{m-3} \ldots b_1) = \mathbf{0}$$
$$\ldots$$
$$req(b_2) \wedge prod(b_1) = \mathbf{0}.$$

Using the addition rule, we have $req(b_i) \wedge prod(b_j) = \mathbf{0}$, where $i, j \in \{1, 2, \ldots, m\}$. Thus, $b_i \perp b_j, i \neq j \in \{1, 2, \ldots, m\}$, which indicates (b).

(b) "$\Rightarrow$" (a) since $b_i \perp b_j, i \neq j \in \{1, 2, \ldots, m\}$, then $req(b_i) \wedge prod(b_j) = \mathbf{0}$, where $i, j \in 1, 2, \ldots, m$, therefore, any orders of $\{b_1, b_2, \ldots, b_m\}$ are feasible. Hence (a) holds. ■

Therefore, an independent relationship can be identified by checking each pair of the sequences $\{\xi_i, \xi_j\}, i \neq j \in \{1, 2, \ldots, n\}$. If there is a common subsequence of ξ_i and $\bar{\xi}_j$, where $\bar{\xi}_j$ is a reversed ξ_j, actions in this common subsequence are recognized as independent actions.

3) Algorithm: We propose the algorithm shown in Algorithm 1. It takes as input Ξ, the set of action sequences labeled from demonstrations, and outputs a table T, which stores the information about meta-actions. A meta-action is a fake action by compacting atomic actions or meta-actions into one, which

Authorized licensed use limited to: Univ of Calif Berkeley. Downloaded on June 19,2021 at 12:24:28 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: Sequential/Parallel Task Model Construction.

Input Ξ
Output T

1: init $T = \emptyset$
2: **while** True **do**
3: $\psi =$ findLCS(Ξ)
4: **if** $|\psi| > 1$ **then**
5: create meta-action A_m
6: Ξ.refresh(A_m)
7: T.update(A_m,relation='s',ψ)
8: **end if**
9: **for** each pair $\{\xi_i, \xi_j\} \in \Xi, i \neq j$ **do**
10: $\psi =$ findLCS(reverse(ξ_i), ξ_j)
11: **if** $|\psi| > 1$ **then**
12: create meta-action A_m
13: Ξ.refresh(A_m)
14: T.update(A_m,relation='i',ψ)
15: **end if**
16: **end for**
17: **if** T is not updated in this iteration **then**
18: break
19: **end if**
20: **end while**
21: **for** each entry t in T with relation='i' **do**
22: $A_o =$ retrieveAtomicActions(T, t.ψ)
23: **if** actions in A_o share no objects of interaction **then**
24: update t.relation='p'
25: **end if**
26: **end for**

corresponds to the intermediate node in the model. Line 1 initializes the table T to be empty. Line 2–20 constructs a hierarchical tree from bottom to top by iteratively discovering the sequential and independent relationships. Line 3–8 identifies the sequential relationship. First, the longest common subsequence ψ is found. If the length of ψ is greater than 1, the actions in subsequence are compacted to a meta-action A_m, subsequence ψ is replaced with A_m for every sequence in Ξ by the function *refresh*, and in the meantime repeated sequences in Ξ are discarded. Finally, the table T is updated by adding an entry of A_m with information of the relation type and the replaced subsequence ψ, where relation types 's,' 'i' and 'p' denotes sequential, independent and parallel relationships. Line 9–16 identifies the independent relationships. The function *refresh* works differently in Line 13. Replacement takes place not for all sequences but for those sequences that contain the subsequence ψ or varying orders of subsequence ψ. The procedure of finding sequential and independent relationships alternates until T table remains the same. Finally, Line 21–26 discovers parallel relationships by checking the objects of interaction among independent actions. Function *retrieveAtomicActions* retrieves all the atomic actions that a meta-action represents. Fig. 3 shows the T table and the updating demonstration set for the desktop assembly task when running this algorithm.

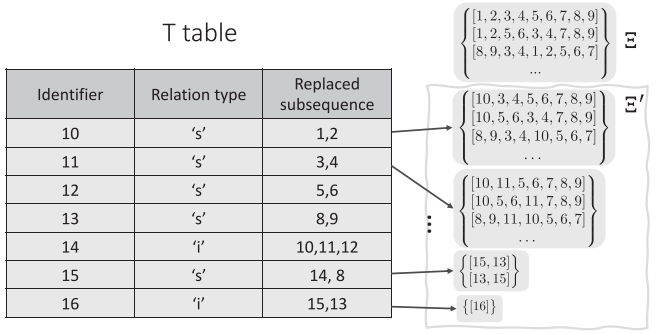

Fig. 3. T table and the updating demonstration set $\Xi'$for the desktop assembly task when running algorithm 1. Action identifier 1-9 corresponds to the atomic action 'Obtain CPU fan,' 'Insert CPU fan,' 'Obtain memory,' 'Insert memory,' 'Obtain tape,' 'Wrap cables,' 'close hood,' 'Obtain label' and 'Apply label'.

4) Demonstration Set Requirement: Our algorithm does not require that the demonstration set to be complete for successfully constructing the task model. Suppose the number of independent nodes in the sequential/parallel task graph is p, and the number of children for ith independent node is $l_i \in \mathcal{N}$ for $i = \{1, 2, \ldots p\}$. If $p \geq 1$, which is often true, the minimum number of different plans required by our algorithm is 2, while the number of all possible plans is $\prod_{i=1}^{p} l_i$. Since $l_i \geq 2$ and $p \geq 1$, thus $2 \leq 2^p \leq \prod_{i=1}^{p} l_i$ holds, indicating that complete demonstration set is not a must for our algorithm.

5) Complexity: The dominating complexity factor throughout the algorithm is the discovery of the independent relationship. The step of finding the longest common subsequence can be accomplished in time $\mathcal{O}(m\log(m))$ using the algorithm in [31], where m is the length of the sequence, in our case the number of actions in a demonstration. Therefore, the time complexity of our algorithm is $\mathcal{O}(mn^2\log(m))$, where n is the number of demonstrations.

B. A Robot Planner

In this section, we will present a human-aware robot task planner. For simplicity, we assume that there is one robot and one human. However, with a slight modification, the algorithm can be applied to a team of multiple humans and multiple robots. The input of the algorithm includes sequential/parallel task model T, the execution indicator s, capability of the human C_h and capability of robots C_r, which is a set of actions that the human or the robot is capable of doing. The planner optimizes a plan, and then sends the robot first action to the controller to execute, and plans again, repeatedly.

The key of our planner is the planning horizon, which is a collection of subtasks including: 1) the sequential subtask the human is currently conducting, and 2) the subtasks that are parallel to the human's current subtasks and are not executed yet, which is illustrated in Fig. 4. Thus, the planning horizon is part of the remainging task and it varies as the task proceeds. Since humans follow abstract hierarchies while doing a task, it is very likely that the human conducts the following actions in the sequential subtasks after finishing the current action, thus we assign the whole sequential subtask to the human to avoid that

Authorized licensed use limited to: Univ of Calif Berkeley. Downloaded on June 19,2021 at 12:24:28 UTC from IEEE Xplore. Restrictions apply.

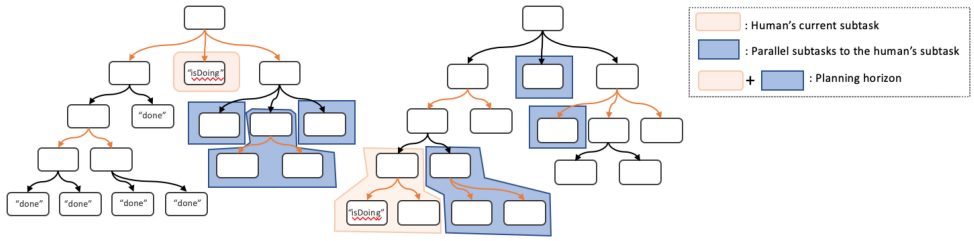

Fig. 4. Two examples of planning horizons. These are two sequential/parallel task model. Red edges are with sequential relationships and black edges are with independent relationships. "done" and "isDoing" in the atomic action node represents that the action is executed already and that the action is being executed by the human. The subtasks with orange shades are the human's current subtasks, and the the subtasks with blue shades are the parallel subtasks to the human's current subtasks. These two kinds of subtasks constitute the planning horizon.

the robot chooses the same action as the human's. The planning problem then becomes a scheduling problem, which is to assign those parallel subtasks to the human and the robot such that the completion time for the planning horizon is minimized. The optimization problem is formulated as follows.

$$\begin{aligned} \min_{x_h, x_r} \quad & t \\ \text{s.t.} \quad & x_h^T t_1 + t_o \leq t \\ & x_r^T t_2 \leq t \\ & x_h + x_r = \mathbf{1} \\ & x_h, x_r \in \{0,1\}^k \\ & x_{h\{C-C_h\}} = 0 \\ & x_{r\{C-C_r\}} = 0 \end{aligned} \tag{1}$$

Decision variables x_h and $x_r \in \{0,1\}^k$ are binary vectors for the assignment of subtasks, where k is the number of subtasks in the planning horizon. Objective function t is the completion time for the planning horizon, which is the upper bound of the human's completion time and the robot's completion time. t_o is the remaining time for the human to complete his current subtask. t_1 and t_2 are the empirical completion time for subtasks, which are obtained based on *Attribute* of an action. $\{C - C_h\}$ denotes the indices of the actions that the human is unable to do, and $\{C - C_r\}$ denotes the indices of the actions that the robot is unable to do.

This planner is suitable to be integrated into a robot system which has two modes of interaction. One mode is "command" mode, where the human explicitly asks the robot to perform an action through communication. The other mode is "automation" mode. The human and the robot collaboratively do tasks without communication. Our planner can work under this "automation" mode, and it is switched to the other planner when humans make commands.

IV. Simulations and Experiments

A. Three Scenarios

We use the following three different scenarios to decrease the impact of different tasks for a fair comparison.

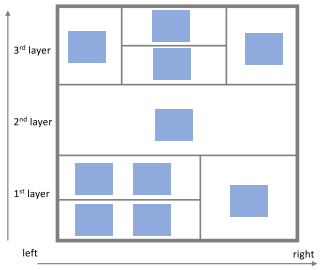

Fig. 5. The book shelving scenarios. The solid boxes represent books, and the gray frames constitute the bookshelf.

1) A Desktop Assembly Scenario: We evaluate our proposed planning algorithm in a desktop assembly task whose sequential/parallel task model is shown in Fig. 1.

2) Two Book Shelving Scenarios: We device two book shelving scenarios, corresponding to the two models in Fig. 4. The initial state of the task is that the shelf is empty, and the goal state is shown in Fig. 5. Difference of the two scenarios is the constraints on the shelving order. For first scenario, books must be shelved from bottom to top, which means that books must be shelved at 1st layer first and then 2nd layer and finally 3 rd layer. This constraint still holds to small layers such as the left two layers at the 1st layer. This task corresponds to the left model in Fig. 4. For the second scenario, there is no constraint on the vertical axis, however, within each layer, books must be shelved from left to right. This corresponds to the right model in Fig. 4.

B. Hypothesis

We evaluate the effectiveness of the proposed planing algorithm through simulations by verifying the following four hypotheses.

- H1: The proposed algorithm generates efficient plans.
- H2: The proposed algorithm generates safe plans.
- H3: The performance of the proposed algorithm is robust to different levels of incompleteness in demonstrations.
- H4: The human subjects are more satisfied with our collaborative robot planning than with other baseline planners.

C. Simulation Setup

We test our algorithm on a simulator built in MATLAB. Human subjects manipulate the objects by using a mouse, dragging and dropping the objects with mouse bottom held down and

Authorized licensed use limited to: Univ of Calif Berkeley. Downloaded on June 19,2021 at 12:24:28 UTC from IEEE Xplore. Restrictions apply.

released. Complex actions such as inserting and wrapping in the desktop assembly scenario are simplified by dropping the objects. The robot actions are expressed by object moving and a sentence shown in the interaction window. The capability of both the human and the robot is set to the whole action space for all the tasks.

Six human subjects participated in the simulation. After instructed about the task and the simulation environment, they operated each action for practice, and that was when the completion time of actions for humans were collected. For robots, the completion time of actions are manually set in the simulations.

D. Manipulated Variables

To evaluate the effectiveness of the proposed algorithm, we manipulated two controlled variables in our simulations. The first one is *planing schemes*. Our algorithm is compared with the following three baselines:

1) Random policy: This planner models the task as a set of action sequences, which is a single-level plan library. It tries to recognize the human's plan by aligning the current procedure/ action sequence to the action sequences in the plan library, and chooses the most likely action sequences as the human's intended plans, which can be many. After that, the robot action is generated uniformly at random from the set of next actions of those possible human's plans.
2) Anticipatory planning [13]: This algorithm models the human's and the robot's behavior with an MDP and learns the policy from demonstrations by reinforcement learning. At the execution time, it also models that human may not take the optimal action by predicting the ratio of the human taking optimal action and human following habits.
3) Shortest completion time optimizer: Similar to our algorithm, this algorithm also aims to minimize the completion time, but the planning horizon is the whole task to go.

The other manipulated variable is *level of incompleteness* of the demonstration set. The levels are set to be 0%, 30%, 50% which means these percentages of action sequences in the demonstration set will be randomly dismissed. Since the third baseline assumes the task model is already known, it does not rely on the demonstration set. Thus, by manipulating the two variables, we have 10 groups of simulations for each scenario. Under each group, every human subject performs the task using any plan for twice. Thus, there will be 20 trials in each simulation group and in total 120 trials for each scenario.

To have a fair comparison, all the algorithms use the same sets of action sequence as inputs under all the incompleteness conditions for each scenario. For the hyperparameters in the algorithm of anticipatory planning, grid search cross-validation is utilized to choose the hyperparameters that perform the best.

E. Dependent Measures

To quantify the safety of the proposed framework, we measure the percentage of robot actions that conflicts with the human's. For efficiency, we set a timer to keep track of the task completion time. The timer starts when a human subject starts to act, and ends when the task is finished. As for the measurement of human's satisfaction with our collaborative robots, we ask the six human subjects to rate the following statements on Likert scale from 1 (strongly disagree) to 5 (strongly agree), similar to [13]: 1. The robot was collaborative and helped; 2. The robot did the right thing at the right time; 3. I am satisfied working with the robot; 4. I will work with this robot again in the future.

F. Results

H1: Table II shows the average task completion time for the three scenarios using different planners. Our planner is significantly more efficient than the planners of random policy and anticipatory planning under all situations[1] ($p < 0.01$). Compared with shortest completion time optimizer, our planner can achieve similar results, as the average task completion time is close and there is no significant difference among all trials ($p > 0.1$). Possible reasons are that: 1) although shortest time optimizer computes the most efficient plan, the human subject may not follow that plan; 2) shortest time optimizer does not consider humans' preference of doing the next actions with a sequential relationship. Thus, this planner is more likely to cause conflicts, which increases the task completion time. Random policy is most inefficient. Opposite to our algorithm, random policy has no knowledge of the parallel relationships, and it strictly plans the next actions of the current plan, which results in a lot of waiting time. Compared with random policy, anticipatory planning has shorter task completion time, as it enables the robot to anticipate more in the collaboration. However, its limitation is that it utilizes learned policies from single-agent demonstrations. In the two-agent scenarios, humans can follow a different reward model, which makes the learned policies less effective. Although this planner tries to adapt to the human on the fly, it takes time.

H2: The average percentages of conflicts are 2.7 ± 0.2 %, $15.0 \pm 0.7\%$, $8.1 \pm 0.4\%$ and $6.2 \pm 0.2\%$ for our algorithm, random policy, anticipatory planning and shortest completion time optimizer correspondingly. Our planner is the safest planner among the other planners, as the robot conflicts with the human the least with significant difference ($p < 0.01$). The planner guides the robot to do subtasks that are parallel to the human's current action, which naturally decreases the possibility of conflicts. This result also validates the assumption stated in Section III-A that humans complete sequential actions consecutively without switching to another parallel subtask. The planner with random policy is the most unsafe one, as it causes conflicts the most with significant difference ($p < 0.01$). This planner has no knowledge of the parallel relationships and it always plans the next actions of the current recognized plan. However, when humans are conducting subtasks with sequential actions, they also prefer to do the next actions, which can cause conflicts. The planners with anticipatory planning and shortest completion time optimizer do not explicitly utilize parallel information, nor do they explicitly utilize sequential information, and their results are in-between.

[1]We use paired t-test for all the statistical tests.

Authorized licensed use limited to: Univ of Calif Berkeley. Downloaded on June 19,2021 at 12:24:28 UTC from IEEE Xplore. Restrictions apply.

TABLE II
THE RESULT TABLE FOR THE AVERAGE TASK COMPLETION TIME (UNIT: SECOND). M1, M2 AND M3 ARE METHODS OF RANDOM POLICY, ANTICIPATORY PLANNING AND OPTIMIZATION RESPECTIVELY

Incompleteness	Desktop assembly Scenario				book shelving scenario 1				book shelving scenario 2			
	ours	M1	M2	M3	ours	M1	M2	M3	ours	M1	M2	M3
0%	**8.3**	12.6	10.0	8.5	16.6	23.7	18.9	**15.5**	**13.2**	20.6	17.7	14.1
30%	**7.5**	15.0	13.4	-	**15.7**	25.2	21.7	-	**14.5**	27.4	25.3	-
50%	**8.1**	15.9	15.8	-	**17.5**	28.1	27.5	-	**15.1**	28.0	27.4	-

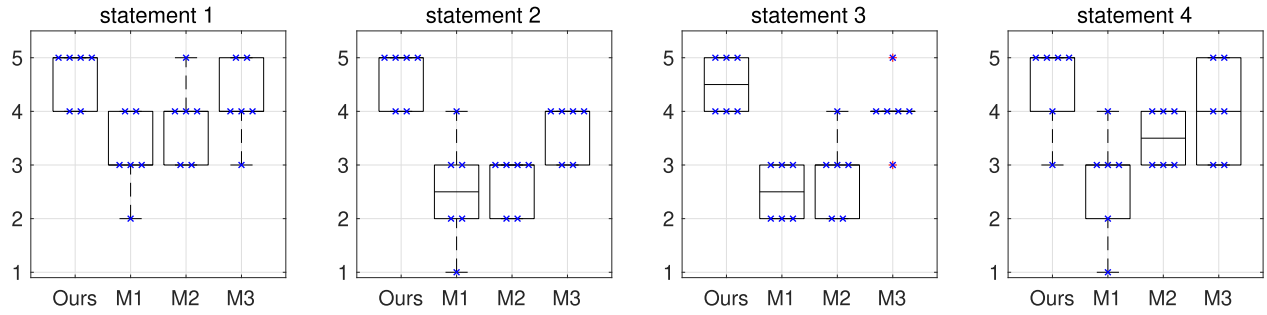

Fig. 6. Human subjects' ratings for the four robot planners on four different statements. M1, M2 and M3 are for planners of random policy, anticipatory planning and optimization correspondingly. Score 1 to 5 corresponds to 1. strongly disagree, 2. disagree, 3. neutral, 4. agree and 5. strongly agree.

H3: Through various simulations on different levels of incompleteness of demonstrations, we show that our method consistently delivers comparable performance as shown in Table II. The average task completion times do not increase as the level of the incompleteness increases. For example, when 30% of the demonstrations are deleted from the desktop assembly scenario, the average task completion time even drops by 0.8 seconds. Besides, through statistical tests, we find that there is no significant difference in the task completion time and percentage of conflicts among all levels of incompleteness for each scenario ($p < 0.01$). Actually, the sequential/parallel task models retrieved from the demonstration set are the same for all the tasks with different incompleteness levels. Thus, our algorithm is robust to the incomplete demonstrations.

H4: Fig. 6 shows the comparison of human subjects' ratings for the four types of planners on the four criteria: 1. The robot was collaborative and helped; 2. The robot did the right thing at the right time; 3. I am satisfied working with the robot; 4. I will work with this robot again in the future. Human subjects rated the our planner significantly higher than the other planners on all four criteria ($p < 0.01$). This indicates that human subjects are more satisfied with our planner. It is also interesting to note that scores of the anticipatory planning are significantly higher than that of random policy ($p < 0.01$), and scores of shortest completion time optimizer are significantly higher than that of anticipatory planning ($p < 0.01$).

G. Experiments

We test our planner on an industrial robot FANUC LR Mate 200iD/7 L. A Kinect V2 for windows is placed close to the table for detecting the human and the objects. Fig. 7 shows that the robot is collaborating with the human to complete the desktop assembly task, where the human is assembling the CPU fan, while the robot is going to assemble the memory based on the plan computed by our planner. The video of the experiment is available at: https://msc.berkeley.edu/research/serocs.html.

Fig. 7. Robot and human collaborate to assemble a desktop.

V. CONCLUSION

To solve the collaborative task planning problem, we first propose an algorithm to automatically construct a hierarchical task model from human demonstrations, which captures the sequential and parallel relationships of the task at all levels of abstraction. This algorithm is very robust to the levels of incompleteness of demonstration. Actually, for any task, two specific demonstrations are enough to retrieve the sequential/parallel task model. We then propose an optimization-based planner, which exploits the parallel relationships in the task model and gives priorities to the actions that are parallel to the humans,' since conducting parallel actions can reduce spatial interfaces, reduce task completion time and improve human's satisfaction. These benefits are verified through various simulations for three different scenarios when compared with other three baselines.

Authorized licensed use limited to: Univ of Calif Berkeley. Downloaded on June 19,2021 at 12:24:28 UTC from IEEE Xplore. Restrictions apply.

We can slightly modify our proposed algorithm for planning with multiple objectives, such as improving safety while reducing task completion time. The solution to this multi-objective problem is optimization over our proposed planning horizon by using methods such as weighted sum method, $\epsilon-$constraint method, weighted metric method, which could be our future work.

REFERENCES

[1] S. Nikolaidis and J. Shah, "Human-robot teaming using shared mental models," *ACM/IEEE HRI*, 2012.

[2] E. Coupeté, F. Moutarde, and S. Manitsaris, "Gesture recognition using a depth camera for human robot collaboration on assembly line," *Procedia Manuf.*, vol. 3, pp. 518–525, 2015.

[3] Y. Cheng, L. Sun, C. Liu, and M. Tomizuka, "Towards efficient human-robot collaboration with robust plan recognition and trajectory prediction," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 2602–2609, Apr. 2020.

[4] Y. Cheng, W. Zhao, C. Liu, and M. Tomizuka, "Human motion prediction using semi-adaptable neural networks," in *Proc. IEEE Amer. Control Conf*, 2019, pp. 4884–4890.

[5] 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. 2016.

[6] S. M. LaValle, *Planning Algorithms*. Cambridge Univ., Press, 2006.

[7] 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.

[8] M. Cirillo, L. Karlsson, and A. Saffiotti, "Human-aware task planning for mobile robots," in *Proc. Int. Conf. Adv. Robot*., 2009, pp. 1–7.

[9] S. J. Levine and B. C. Williams, "Concurrent plan recognition and execution for human-robot teams," in *Proc. 24th Int. Conf. Automated Plan. Scheduling*, 2014.

[10] W. Wang, R. Li, Z. M. Diekel, and Y. Jia, "Robot action planning by online optimization in human-robot collaborative tasks," *Int. J. Intell. Robot. Appl.*, vol. 2, no. 2, pp. 161–179, 2018.

[11] M. Cirillo, L. Karlsson, and A. Saffiotti, "Human-aware task planning: An application to mobile robots," *ACM Trans. Intell. Syst. Technol.*, vol. 1, no. 2, pp. 1–26, 2010.

[12] V. Montreuil, A. Clodic, M. Ransan, and R. Alami, "Planning human centered robot activities," in *Proc. IEEE Int. Conf. Syst., Man Cybern*., 2007, pp. 2618–2623.

[13] H. S. Koppula, A. Jain, and A. Saxena, "Anticipatory planning for human-robot teams," in *Exp. Robot*. Berlin, Germany: Springer, 2016, pp. 453–470.

[14] R. A. Knepper, D. Ahuja, G. Lalonde, and D. Rus, "Distributed assembly with and/or graphs," in *Proc. Workshop AI Robot. Int. Conf. Intell. Robots Syst.*, 2014.

[15] T. Boyd, M. Zagainova, J. Blankenburg, and M. N Nicolescu, "Hierarchical Task Learning Through Human Demonstration," 2019.

[16] A. Roncone, O. Mangin, and B. Scassellati, "Transparent role assignment and task allocation in human robot collaboration," in *Proc. IEEE Int. Conf. Robot. Automat*, 2017, pp. 1014–1021.

[17] B. Hayes and B. Scassellati, "Autonomously constructing hierarchical task networks for planning and human-robot collaboration," in *Proc. IEEE Int. Conf. Robot. Automat*., 2016, pp. 5469–5476.

[18] M. C. Gombolay, R. J. Wilcox, and J. A. Shah, "Fast scheduling of robot teams performing tasks with temporospatial constraints," *IEEE Trans. Robot.*, vol. 34, no. 1, pp. 220–239, Feb. 2018.

[19] 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.

[20] A. Garland, K. Ryall, and C. Rich, "Learning hierarchical task models by defining and refining examples," in *Proc. 1st Int. Conf. Knowl. Capture*. ACM, 2001, pp. 44–51.

[21] G. Mori, F. Paternó, and C. Santoro, "CTTE: Support for developing and analyzing task models for interactive system design," *IEEE Trans. Softw. Eng.*, vol. 28, no. 8, pp. 797–813, Aug. 2002.

[22] S. Qi, S. Huang, P. Wei, and S.-C. Zhu, "Predicting human activities using stochastic grammar," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1164–1172.

[23] M. Pei, Y. Jia, and S.-C. Zhu, "Parsing video events with goal inference and intent prediction," in *Proc. IEEE Int. Conf. Comput. Vis*., 2011, pp. 487–494.

[24] J. B. Tenenbaum, C. Kemp, T. L. Griffiths, and N. D. Goodman, "How to grow a mind: Statistics, structure, and abstraction," *Science*, vol. 331, no. 6022, pp. 1279–1285, 2011.

[25] C. Kemp, A. Perfors, and J. B. Tenenbaum, "Learning overhypotheses with hierarchical bayesian models," *Devlop. Sci.*, vol. 10, no. 3, pp. 307–321, 2007.

[26] R. Salakhutdinov, J. Tenenbaum, and A. Torralba, "One-shot learning with a hierarchical nonparametric bayesian model," in *Proc. ICML Workshop Unsupervised Transfer Learn.*, 2012, pp. 195–206.

[27] J. Liu, B. Kuipers, and S. Savarese, "Recognizing human actions by attributes," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 3337–3344.

[28] H. de Almeida Maia *et al.*, "Action recognition in videos using multi-stream convolutional neural networks," in *Deep Learn. Appl*. Berlin, Germany: Springer, 2020, pp. 95–111.

[29] L. Liu *et al.*, "Deep learning for generic object detection: A survey," *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 261–318, 2020.

[30] B.-N. Vo and B.-T. Vo, "A multi-scan labeled random finite set model for multi-object state estimation," *IEEE Trans. Signal Process.*, vol. 67, no. 19, pp. 4948–4963, Oct. 2019.

[31] R. Bhowmick, M. I. S. Bhuiyan, M. S. Hossain, M. K. Hossen, and A. S. Tanim, "An approach for improving complexity of longest common subsequence problems using queue and divide-and-conquer method," in *Proc. 1st Int. Conf. Adv. Sci., Eng. Robot. Technol.*, 2019, pp. 1–5.

Authorized licensed use limited to: Univ of Calif Berkeley. Downloaded on June 19,2021 at 12:24:28 UTC from IEEE Xplore. Restrictions apply.