Interactive Hierarchical Task Learning from a Single Demonstration

Anahita Mohseni-Kabir, Charles Rich, Sonia Chernova, Candace L. Sidner, Daniel Miller
Worcester Polytechnic Institute
Worcester, MA, USA
{amohsenikabir, rich, soniac, sidner, millerd}@wpi.edu

ABSTRACT

We have developed learning and interaction algorithms to support a human teaching hierarchical task models to a robot using a single demonstration in the context of a mixed-initiative interaction with bi-directional communication. In particular, we have identified and implemented two important heuristics for suggesting task groupings based on the physical structure of the manipulated artifact and on the data flow between tasks. We have evaluated our algorithms with users in a simulated environment and shown both that the overall approach is usable and that the grouping suggestions significantly improve the learning and interaction.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics

Keywords

Hierarchical Task Networks; Learning From Demonstration

1. INTRODUCTION

This paper focuses on the problem of how a robot can efficiently learn a hierarchical task model from a human teacher who is an expert in the domain, but not in robot programming. Our approach integrates learning from demonstration (LfD) with hierarchical task networks (HTNs). We use HTNs instead of the flat representations more commonly used in LfD because HTNs are more intuitive and computationally more tractable, especially for non-programmers and complex tasks.

Additionally, our approach consists of viewing LfD as a collaborative discourse (see Figure 1). In a collaborative discourse, both participants (in this case, the human teacher and the robot learner) are committed to the shared goal of the interaction (in this case, for the robot to learn a new task) and both actively contribute towards achieving that goal. More specifically, in the situated interaction illustrated in Figure 1, which is our target, both the human and the robot can manipulate and observe the other's manipulations of artifacts in the shared environment, and there is bi-directional communication. Collaborative discourse theory [5] provides a foundation for both the algorithms and the implementation of our system. More specifically, the rules in this theory for managing the focus of attention (as a stack) have guided our interaction design and we are using an open-source tool, called Disco [14], based on collaborative discourse theory in our implementation.

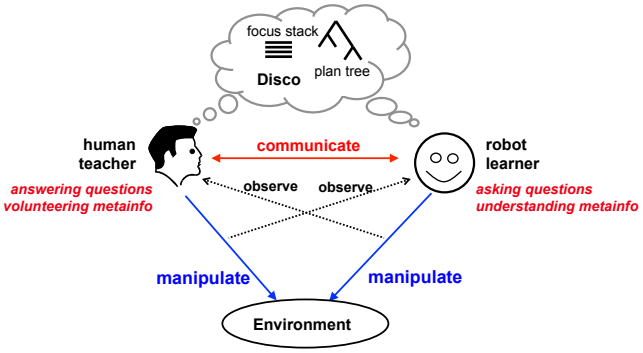


Figure 1: LfD as a collaborative discourse.

Finally, in this paper we focus on learning from a single demonstration. As we will see below, this is possible because of the bi-directional communication between the teacher and the learner. However, sometimes multiple demonstrations are unavoidable. Our other work [10] explores how to merge multiple demonstrations in HTNs.

The example domain we have chosen for this research is maintenance tasks and specifically car maintenance, because it is challenging yet familiar to most readers. We have started with tire rotation and expect in the future to teach a robot how to check the oil, change filters and belts, and so on.

This paper is organized around the running example of a tire rotation HTN, which is introduced in Section 2. In Section 3, we first describe how our algorithms can learn the hierarchical structure of this HTN from a single "ideal" interactive demonstration. However, our user study indicates that most users are far from ideal, which motivates the need for suggestions from the

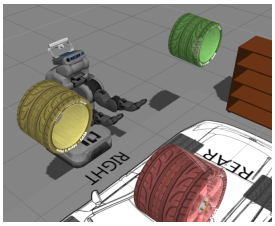


Figure 2: Simulated environment with PR2.

HRI'15, March 2–5, 2015, Portland, Oregon, USA.

ACM 978-1-4503-2883-8/15/03 ...$15.00.
http://dx.doi.org/10.1145/2696454.2696474.

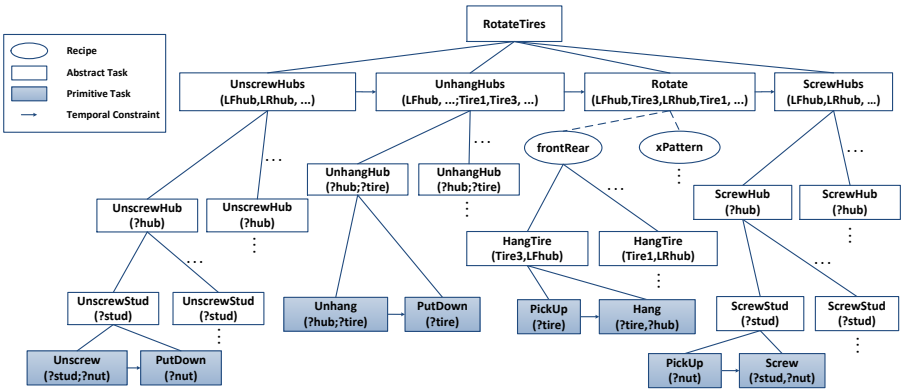


Figure 3: A hand-coded hierarchical task network for tire rotation.

robot, which we then describe. Section 4 describes our user study in detail. It shows both that the overall approach is usable and that the robot's suggestions significantly improve the learning and interaction. We discuss related work in Section 5, after we have presented our results.

The intended application of this work is with a real robot, such as the PR2, in a shared physical environment. As a first step, however, we have implemented and evaluated our techniques using Gazebo in the simulated environment shown in Figure 2. See Section 6 for a description of the physical environment we are building and other next steps.

2. HIERARCHICAL TASK NETWORKS

HTNs are widely used in artificial intelligence, especially in interactive systems. An HTN is essentially a tree in which the fringe nodes denote *primitive* tasks, i.e., tasks that can be directly executed (e.g., by a robot), and the other nodes denote *abstract* tasks, which must be decomposed in order to be executed. As an example, consider the HTN for tire rotation shown in Figure 3. This is a hand-coded version of the kind of HTN that our system learned from the participants in our user study.

The abstract tasks in an HTN are not absolutely necessary to execute the toplevel task—RotateTire could simply be represented as a sequence of 64 primitives. However, there is much evidence that humans naturally think about complex tasks hierarchically. For example, collaborative discourse theory is based on a hierarchy of goals.

The abstract tasks in an HTN provide an important vocabulary for communication between the human and robot in the context of a collaboration. This shared vocabulary is needed for discussing the partial completion status of the task, for delegating subtasks, for discussing potential problems with execution, and so on. Furthemore, the abstract tasks can often be reused in other situations. For example, after learning the HTN in Figure 3, the robot not only knows how to rotate tires, but it has also learned two abstract tasks (UnscrewHub and UnhangHub) that are useful for fixing a flat tire.

HTNs are also commonly used as an alternative to symbolic planning in complex real-world applications where a complete logical formalization is difficult or infeasible. Although tire rotation is simple enough that it could easily be formalized for a symbolic planner, our target is applications in which this is not true. The learning algorithms in this paper therefore require only the name and input types of a task.

There are many different formalisms available for representing HTNs. In this work, we use the ANSI/CEA-2018 standard [13] because, among other things, it explicitly represents data flow between tasks, which is the basis for one of the grouping heuristics described in Section 3.2. In other work [11] we have used data flow in HTNs to learn temporal constraints from a single demonstration.

Since the HTN in Figure 3 will be used as a running example (and was the domain of our user study), it is worthwhile examining it here in some detail. First, Figure 4 shows the terminology we are using for the relevant parts of a car: a tire goes on a hub and is secured with three nuts (one on each stud). The hubs are identified as LF, LR, RF, RR, standing for the left-front, left-rear, etc., and each hub has three studs, identified as "LFhub.stud1", etc. The tires are named Tire1, Tire2, etc., and in the user study have distinct colors for convenience of identification.

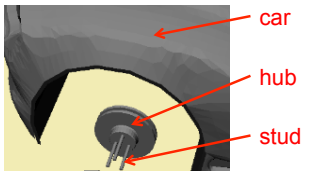


Figure 4: Terminology.

In ANSI/CEA-2018, both primitive and abstract tasks have typed inputs and outputs. The tire rotation HTN uses six primitive task types: Screw, Unscrew, Hang, Unhang, PickUp, and PutDown, with their respective input and output types shown. For example Unscrew (see Figure 5) takes a stud as input and returns a nut as output. There are 64 primitive tasks in the fringe of the tire rotation tree (some of the repetitive structure has been elided in the figure to save space).

Figure 3 defines 11 abstract task types, such as UnscrewHubs, which are used 45 times in the interior of the tree. This particular way of decomposing the tire rotation task is not the only possible way, but the other reasonable decompositions all have a similar number of abstract types and total number of nodes.

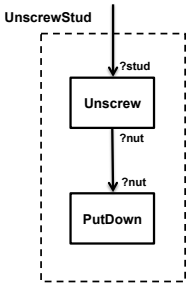


Figure 5: Data flow.

In ANSI/CEA-2018, the subtasks of a given task are by default *unordered*. An explicit temporal constraint can be added between two tasks to specify a partial ordering. For example, the four toplevel steps of RotateTires are totally ordered, but the steps of UnscrewHubs are unordered, since it does not matter in what order you unscrew the nuts.

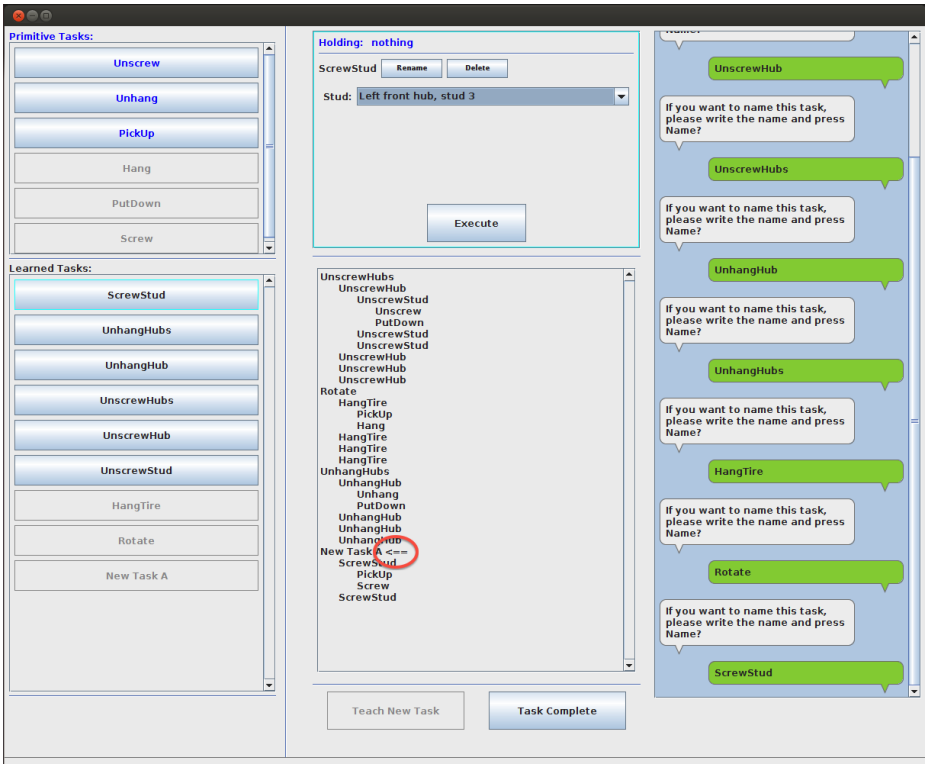


Figure 6: Graphical user interface for teacher.

Data flow in ANSI/CEA-2018 is an identity constraint between a task's output and an input of a sibling task, i.e., another subtask of the same parent task. Figure 5 shows an example of data flow in the tire rotation HTN.

A key feature of HTN's that we are not dealing with in this work is the ability to specify alternate decompositions, or *recipes*. Figure 3 provides two recipes for the Rotate step of tire rotation corresponding to two different rotation patterns: front-to-rear and corner-to-corner ("X-pattern"). Learning alternative recipes is one of those circumstances in which multiple demonstrations are unavoidable. In this paper, however, we are only learning the front-to-rear recipe.

3. LEARNING FROM DEMONSTRATION

In this section we will explain, using two example interactions, how our system learns the hierarchical structure of the tire rotation HTN in Figure 3. The first example interaction, in Section 3.1, is a hypothetical "ideal" interaction in which the robot provides no helpful suggestions, but the user teaches exactly the hand-coded hierarchical structure anyway. While this interaction is fully supported by our system, in our user study, no participants came even close to this ideal behavior when the robot provided no suggestions; instead they made far less optimal task structures. In the second interaction, described in Section 3.2, the same HTN is learned *with* suggestions from the robot that guide the user in structuring the task. This interaction is in fact typical of what we saw with our users who received and accepted suggestions from the robot.

Our focus in this work is on learning the grouping structure of the HTN. The names chosen for abstract tasks and some details of the inputs and outputs may vary from our hand-coded HTN. Furthermore, our current system only supports a bottom-up style of learning, in which primitives are first combined into small abstract tasks, which are then combined into bigger abstract tasks, and so on. This style lends itself well to demonstration. In the future (see Section 6), we plan to extend our techniques to top-down and mixed modes of learning.

Figure 6 shows the graphical user interface we have developed for a human to use with our simulated robot environment. In use, the human teacher interacts with this interface while simultaneously seeing the effects of actions on a side-by-side simulation screen, similar to Figure 2. Figure 6 shows the state of the GUI at line 32 of Figure 7.

Before we describe its operation in detail, we also need to emphasize that this GUI is *not* a research goal in its own right, but rather a means to test our learning algorithms and overall interaction design while still in simulation. As we discuss in Section 6, most or all of this GUI will disappear once we move into a physical shared human-robot environment.

The left side of the GUI contains buttons that select a primitive (top left) or abstract (bottom left) task to execute. The primitive task types are fixed. New abstract task buttons are added whenever a new abstract task type is learned.

The top middle area of the GUI is mainly for specifying inputs, if any, as needed before executing a task. Each input can be selected from a drop-down list of objects in the environment of the correct type. When the user presses the Execute button, the robot executes the selected primitive or abstract task.

The Rename and Delete buttons at the top of this area are for renaming and deleting learned abstract tasks. The Holding line above these buttons is a small informational

display that keeps track of the object, if any, that the robot is currently holding.

The middle area of the GUI below the input selection area is the main informational display, which shows the current hierarchical structure of the learned tasks. The small arrow next to New Task A (red circle added here for visibility) indicates the abstract task currently being learned. This area is also where the highlighting appears for tasks suggested for grouping, as described in Section 3.2. The Teach New Task and Task Complete buttons below this area respectively start and end the demonstration of a new abstract task.

Finally, the right side of the GUI contains a "chat" window in which the robot can ask the user questions and the user can reply. For example, whenever the user presses the Task Complete button, the robot prompts the user for a name for the newly learned abstract task. This is also where the robot's helpful suggestions appear and where the user can accept or reject them.

3.1 Learning Without Suggestions

Figure 7 shows a hypothetical interaction that results in the system learning the hierarchical structure of Figure 3. The user starts by teaching a new abstract task (which will be named UnscrewStud), demonstrating its two primitive subtasks: Unscrew (applied to stud1 of the LFhub) and PutDown (applied to the nut it is holding as a result).

```
1  Teach New Task
2    Execute Unscrew(LFhub.stud1)
3    Execute PutDown(nut11)
4  Task Complete: UnscrewStud
5  Teach New Task
6    Execute UnscrewStud(LFhub.stud2)
7    Execute UnscrewStud(LFhub.stud3)
8  Task Complete: UnscrewHub
9  Teach New Task
10   Execute UnscrewHub(LRhub)
11   Execute UnscrewHub(RFhub)
12   Execute UnscrewHub(RRhub)
13 Task Complete: UnscrewHubs
14 Teach New Task
15   Execute Unhang(LFhub)
16   Execute PutDown(Tire1)
17 Task Complete: UnhangHub
18 Teach New Task
19   Execute UnhangHub(LRhub)
20   Execute UnhangHub(RFhub)
21   Execute UnhangHub(RRhub)
22 Task Complete: UnhangHubs
23 Teach New Task
24   Execute PickUp(Tire3)
25   Execute Hang(Tire3,LFhub)
26 Task Complete: HangTire
27 Teach New Task
28   Execute HangTire(Tire1,LRhub)
29   Execute HangTire(Tire4,RFhub)
30   Execute HangTire(Tire2,RRhub)
31 Task Complete: Rotate
32 ...
33 Teach New Task
34   Execute UnscrewHubs(LFhub,LRhub,RFhub,RRhub)
35   Execute UnhangHubs(LFhub,LRhub,RFhub,RRhub)
36   Execute Rotate(LFhub,Tire3,LRhub,Tire1, ...)
37   Execute ScrewHubs(LFhub,LRhub,RFhub,RRhub)
38 Task Complete: RotateTires
```

Figure 7: An hypothetical ideal interaction to teach tire rotation without robot's suggestions.

Then, in lines 5–8, the user applies this new abstract task to the other two studs of the LFhub. Notice that when the user completes this demonstration and names this new higher-level abstract task UnscrewHub, the system includes the immediately preceding demonstration of UnscrewHub in the abstraction, so that UnscrewHub has *three* subtasks, one for each stud.

Continuing in lines 9–13, the user similarly applies this new abstract task to the other three hubs and names this new yet higher-level abstract task UnscrewHubs (plural). As before, the system includes the immediately preceding demonstration in the abstraction, so that UnscrewHubs has four subtasks, one for each tire. At this point all the nuts are off the car.

Next, in lines 14–22, the user similarly teaches a high-level abstract task, UnhangHubs, which takes off all the tires and leaves them on the ground. The heart of tire rotation is then in teaching the Rotate abstract task in lines 23–31, where the user hangs the tires back on in the rotated pattern (Tire3 on LFhub, etc.).

To save space, 13 button presses (including 7 Execute buttons) are elided at line 32, culminating in the learning of ScrewHubs. The learning process for ScrewHubs is entirely parallel to the learning of UnscrewHubs in lines 1–13. Finally, in lines 33–38, the four highest-level abstract tasks are put together to make RotateTires.

```
1  Teach New Task
2    Execute Unscrew(LFhub.stud1)
3    Execute PutDown(nut11)
4    Robot: "Do you want to group the highlighted steps
5    (Unscrew and PutDown) into a new subtask?"
6    "Yes"
7    Robot: "Should I execute Unscrew/PutDown on other studs
8    of left front hub?"
9    "Yes"
10   Robot: "Should I execute Unscrew/PutDown*3 on other
11   hubs of the car?"
12   "Yes"
13   Execute Unhang(LFhub)
14   Execute PutDown(Tire1)
15   Robot: "Do you want to group the highlighted steps
16   (Unhang and PutDown) into a new subtask?"
17   "Yes"
18   Robot: "Should I execute Unhang/PutDown on other hubs
19   of the car?"
20   "Yes"
21   Execute PickUp(Tire3)
22   Execute Hang(Tire3,LFhub)
23   Robot: "Do you want to group the highlighted steps (PickUp
24   and Hang) into a new subtask?"
25   "Yes"
26   Robot: "Do you want to group the highlighted steps
27   (Unhang/PutDown*4 and PickUp/Hang) into a new subtask?"
28   "No"
29   Execute PickUp/Hang(Tire1,LRhub)
30   Execute PickUp/Hang(Tire4,RFhub)
31   Execute PickUp/Hang(Tire2,RRhub)
32   Robot: "Do you want to group the highlighted steps (four
33   HangTire's) into a new subtask?"
34   "Yes"
35   Robot: "Do you want to group the highlighted steps
36   (Unhang/PutDown*4 and HangTire*4) into a new subtask?"
37   "No"
38 ...
39 Task Complete: RotateTires
```

Figure 8: A typical interaction to teach tire rotation with robot's suggestions.

```
last ← last task executed
if |inputs(last)| ≠ 1 then return
input ← first(inputs(last)); whole ← parent(input)
parts ← ∅
foreach other ∈ type(input)
  if other ≠ input ∧ parent(other) = whole
    then add other to parts
if parts ≠ ∅
  then ask 'Should I execute '+last+' on other '
       +type+'s of '+whole+'?'
```

(a) Parts heuristic

```
last ← last task executed
previous ← task executed immediately before last
foreach input of last
  if source(input) = previous ∧ holding input after previous
    then ask 'Do you want to group the highlighted steps ('
         +previous+' and'+last+') into a new subtask?'
      return
```

(b) Data flow heuristic

Figure 9: Pseudocode for suggestion heuristics.

In summary, in this ideal interaction, the user provided 28 task execution instructions and the robot learned 11 new abstract task types, including the toplevel RotateTires. (The robot, of course, executed more than 28 actions because some of the tasks instructed by the user were abstract.)

Unfortunately, as mentioned earlier, no users in our study came even close to this ideal behavior. The best anyone achieved was 6 new abstract task types. So, the question is, how can we help users get closer to this ideal behavior? The next section describes our approach, which is to use general heuristics to generate helpful suggestions.

3.2 Heuristics for Generating Suggestions

Reflecting on the ideal interaction in Figure 7, we identified two general heuristics to help the robot learn HTNs with useful abstract tasks. Because these are heuristics, which means that they are not always right, we communicate the robot's suggestions as questions to which the user can say yes or no (to accept or reject the suggestion).

The first heuristic, called the *parts* heuristic, is inspired by the grouping of repeated tasks in the ideal interaction, which is driven by the part-whole structure of the domain, as shown in Figure 10. The heuristic rule suggests repeating the last executed task on the other parts of the same type that share the same physical parent (e.g., the other two studs of a hub). If this suggestion is accepted, the system then groups the last task together with repetitions for the other parts into a new abstract task.

The second heuristic, called the *data flow* heuristic, is inspired by the grouping that the ideal user does in lines 1–4 of Figure 7. The heuristic rule suggests grouping two consecutive tasks whenever an output of the first is an input of the second (e.g., the data flow shown in Figure 5) *and* the robot is holding the data flow object between the two tasks.

Figure 9 shows pseudocode for these two heuristics, which are applied in order after the execution of each primitive or abstract task. Notice also that both of these heuristics are quite general. They may be more or less useful in different domains, but they are expressed entirely in terms of mathematical concepts, such as data flow and part-whole relationships. There is nothing specifically about car maintenance in the statement of the rules.

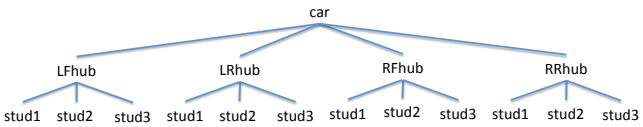


Figure 10: Part-whole knowledge.

Now let's see how these heuristics function in the typical interaction shown in Figure 8. The first suggestion, based on the data flow heuristic, appears on line 4. The user accepts the suggestion, leading to the creation of a new abstract task type called Unscrew/PutDown. Notice that system automatically generates a convenient default name for new abstract task types created in response to suggestions. The user can rename these using the Rename button at the top middle of the GUI.

The second suggestion, based on the parts heuristic, appears on line 7 and results in the new abstract task Unscrew/Putdown*3, which corresponds to UnscrewHub in Figure 7.

The interaction proceeds now in a parallel fashion to Figure 7, with the user answering yes to the next three suggestions. Then, on line 28, the user first answers no. This suggestion is generated by the data flow heuristic (in this case applied to abstract tasks). Because the user has just executed the first step of the front-back tire rotation pattern, which is to put Tire3 (that was originally on the LRhub) onto the LFhub, it seems plausible that she already has in mind the idea of grouping this with the next three steps, not the previous task.

The next suggestion, appearing on line 32, suggests in fact grouping these steps into a new abstract task, which corresponds to Rotate in Figure 7. The user accepts this suggestion.

The second instance in which the user answers no appears on line 37, in answer to another suggestion generated by the data flow heuristic. This is a case in which a yes answer would also be reasonable. All that answering yes would do is to add an extra layer of grouping near the top of the hierarchy, grouping together what are called UnhangHubs and Rotate in Figure 3.

To save space, the remaining three suggestions are elided. They follow the same pattern as the first three suggestions, just as the elided lines in Figure 7 follow the same pattern as lines 1–13 in that interaction.

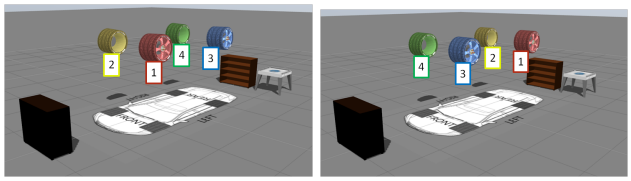
In summary, notice that in this interaction the user executed only 11 tasks, as compared to 28 in Figure 7. Even with answering 12 yes/no questions, the user in this interaction is expending less effort to achieve the same quality plan. This effect is borne out by the user study described in the next section.

4. EVALUATION

We conducted a study with non-expert users to evaluate both the usability of our algorithms in general, and more specifically, the contribution of the suggestions toward improving the interaction and learning.

4.1 Study Design

We conducted a between-subjects study to evaluate two conditions: a *No-Suggestions* condition, in which the robot provided no repetition or grouping suggestions, and a *Suggestions* condition, in which these suggestions were made when



(a) Start configuration (b) Goal configuration

Figure 11: Start and Goal Configurations

appropriate. Participants in both conditions were able to use the GUI to add hierarchy to their task themselves, as in Figure 7.

A total of 32 college-age participants (18 female and 14 male) were recruited. Most of the participants (15 female and 8 male) did not have programming experience. 15 participants were assigned to the No-Suggestions condition and 17 to the Suggestions condition. The two conditions were balanced for gender and programming experience.

4.2 Task

The study consisted of a training activity (in a blocks world domain) followed by the main study based on tire rotation. All participants were given the same training activity to develop familiarity with the GUI and the overall interaction style. Specifically, participants were given step-by-step written instructions for building a tower using six colored blocks and two primitive actions: Pickup and PutOn. Participants were guided through a series of steps, beginning with simply moving blocks around, to teaching a new abstract task, to reusing already learned abstract tasks to build a tower. The training process did not include any robot suggestions, to avoid biasing participants in the No-Suggestions condition. All of the participants successfully completed the training steps and did not have any questions for the experimenter afterwards.

After training, participants performed the main study activity in which they were asked to teach the robot how to rotate the tires from the start configuration to the goal configuration shown in Figure 11. Participants were given detailed written descriptions of the six primitive actions described in Section 2 and a terminology picture similar to Figure 4, but no step-by-step instructions on how to perform tire rotation. The GUI used by participants in both conditions was identical. The only difference between the two conditions was whether or not repetition and grouping suggestions were provided.

4.3 Procedure

Upon arrival, participants were asked to sign an informed consent form and were surveyed about their programming experience. Following this, participants were given the written instructions for the training activity; the experimenter started the learning system and left the room until training was complete. Following training, the experimenter provided the written tire rotation instructions, started the system and again waited outside the room. Upon completion of the tire rotation task, participants were asked to fill out a questionnaire about their experience. Participants were then asked to teach the tire rotation task a second time (under the same study condition), followed by the same questionnaire.

We had participants perform two trials of the main study task because we anticipated that improved performance would occur as they gained experience with the GUI. As we will see in the results below, we did observe this effect.

4.4 Measures and Analysis

We used the following three objective measures to evaluate participant performance:

- *Teaching Effort:* The effort expended by the human, measured as the number of times the teacher pressed the Execute button during the interaction (for both primitive and abstract tasks). We think of this as the number of task instructions communicated by the human to the robot.
- *Plan Quality:* The quality of the learned tire rotation plan, measured as the number of abstract tasks learned (and used). We choose this measure because hierarchy is a valuable property in complex plans for both communication and reuse.
- *Teaching Efficiency:* The efficiency of the interaction, measured by dividing plan quality by the teaching effort (using the definitions above).

The two-tailed Kolmogorov-Smirnov test was used for the evaluation of these measures.

4.5 Hypotheses

We formed the following five hypotheses about the effect of suggestions on the learning interaction based on the models we presented earlier and findings from human-computer and human-robot interaction studies.

- **Hypothesis 1:** Participants will expend less teaching effort in the Suggestions condition than in the No-Suggestions condition.
- **Hypothesis 2:** The quality of the learned plans will be higher in the Suggestions condition than in the No-Suggestions condition.
- **Hypothesis 3:** The teaching efficiency will be greater in the Suggestions condition than in the No-Suggestions condition.
- **Hypothesis 4:** In the Suggestions condition, participants who accepted more suggestions will have better measures of teaching effort and plan quality.
- **Hypothesis 5:** All participants in both conditions will be able to successfully teach the tire rotation task by the second trial.

4.6 Results and Discussion

Table 1 and Figure 12 summarize the results of our study, which support Hypotheses 1–4, but not Hypothesis 5.

Teaching Effort: Analysis of the number of Execute button presses during the training process shows that the teaching effort was significantly higher in the No-Suggestions than in the Suggestions condition, supporting Hypothesis 1. Notice that the p-value in the second trial is lower than in the first trial due to the large reduction in Execute button presses by participants in the Suggestions condition during the second trial. We attribute this difference to participants adapting to the robot's suggestions. A reduction in effort was also

$Mean \pm SD$	Execute Button Presses (First Trial)	Execute Button Presses (Second Trial)	Abstract Tasks Learned (First Trial)	Abstract Tasks Learned (Second Trial)	Efficiency (First Trial)	Efficiency (Second Trial)
No-suggestions	62.07 ± 19.10	51.40 ± 27.22	2.87 ± 2.83	2.47 ± 1.77	0.04 ± 0.03	0.06 ± 0.06
Suggestions	50.47 ± 42.68	26.65 ± 16.42	7.94 ± 5.48	8.53 ± 3.92	0.31 ± 0.31	0.41 ± 0.30
p-value	0.05	0.01	0.02	$\ll 0.001$	$\ll 0.001$	0.001

Table 1: Comparison of objective measures between the two conditions.

observed in the No-Suggestions condition, but the change is much smaller. To further highlight the impact of suggestions on teaching effort, Figure 12(a) shows via linear regression how the number of Execute button presses in both trials decreases with the number of robot suggestions accepted by the user, partially supporting Hypothesis 4.

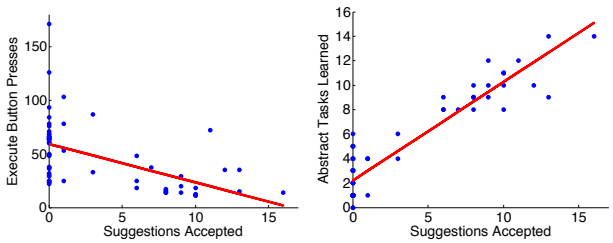
Plan Quality: Analysis of the number of abstract tasks learned shows that the quality of plans was significantly higher in the Suggestions than in the No-Suggestions condition, supporting Hypothesis 2. Additionally, Figure 12(b) shows via linear regression how the number of abstract tasks learned in both trials increases with the number of robot suggestions accepted by the user, further supporting Hypothesis 4.

Teaching Efficiency: Analysis of the efficiency measure provides insight into how teaching effort and plan quality vary in combination between conditions. Teaching efficiency (see last column of Table 1) is significantly higher in the Suggestions than in the No-Suggestions condition, supporting Hypothesis 3. Additionally, we observe greater improvement in efficiency between the first trial and the second trial for participants in the Suggestions condition.

Hypothesis 5 predicted that by the second trial all participants would successfully teach the tire rotation task. Unfortunately, this hypothesis was not supported by our study. In the second trial, for 3/15 participants in the No-Suggestions condition and 4/17 in the Suggestions condition, even though the world state was in the goal configuration at the end of their teaching session, the learned plan would not successfully achieve the goal configuration from the start configuration specified in Figure 11. This is a problem we are looking to solve in the next iteration of our system.

Finally, it is interesting to note that in informal post-study debriefing several participants in the Suggestions condition commented that they initially rejected the robot's suggestions because there had been no suggestions in the training activity and they were were not sure about what the effect of saying yes would be. This was an unintended side effect of our study design, in which we sought to eliminate bias by following the same training procedure in both conditions. Nevertheless, the Suggestions condition showed clear advantages over No-Suggestions despite this limitation, and we expect better results could have been achieved with a training process tailored for the Suggestions condition.

We did not see any significant effect of programming experience on our results.



(a) Teaching effort (b) Plan quality

Figure 12: Impact of accepted suggestions.

5. RELATED WORK

There is extensive research (involving both robots and virtual agents) on learning from demonstration [1], hierarchical task learning, interactive task learning and learning from a single task iteration. However, other than Rybski et al., discussed below, we do not know of any other work that combines all of these aspects, as our present work does.

Research on hierarchical task learning includes both learning from demonstration and learning from instructions approaches. LfD approaches generally focus on learning from multiple demonstrations. For example, in robotic research, Nicolescu and Mataric [12] implemented a LfD system that enables a robot to learn and refine representations of complex tasks and to generalize multiple demonstrations into a single graph-like representation. In [16], Veeraraghavan and Veloso developed a similar LfD approach for teaching a robot tasks with repetitions. Hayes and Scassellati [6] learn HTNs through multi-agent coordination with humans in the loop.

Garland et al. [4] developed an offline, purely software system that generalizes an HTN from a set of execution traces (demonstrations). A novel aspect of their system was support for a domain expert to refine past demonstrations. In addition to relying on multiple demonstrations, this system does not make any suggestions—the user must take all of the initiative in building the HTN.

Most research on interactive task learning goes under the rubric of "active learning" and involves the system asking questions of the user. We prefer the term "interactive" to emphasize the bidirectional nature of the communication shown in Figure 1.

For example, Chernova and Veloso [3] developed a policy learning algorithm in which the robot asks the human to provide labels for states in which the robot has a low certainty. Focusing more on social interaction and broader question types, Cakmak and Thomaz [2] have formalized three question types—label, demonstration and feature requests—and studied their use in LfD. Hayes and Scassellati [7] also generate demonstration requests. None of these systems, however, generate suggestions for grouping.

The most common approach to learning from a single task iteration is via instructions. For example, Huffman and Laird [8] explored the requirements for designing an instructable agent, and showed that a simulated robotic agent could learn useful instruction in natural language. In [9], Mohan and Laird extended this work by instantiating their design in the Soar cognitive architecture. Their work uses

an HTN as the task representation, but does not include demonstrations.

The closest work to ours is by Rybski et al. [15], who developed an algorithm that combines spoken language understanding, dialog and physical demonstration to learn complex plans. Their task representation allows abstract tasks to be constructed from simpler subtasks in order to create hierarchical structures. The robot can also verify the HTN with the human by asking questions and allowing the human to add additional conditional cases. However, their approach for learning task structure differs from ours in that it does not leverage ordering and data flow constraints. Furthermore, the robot's questions are limited to filling in unspecified conditions (e.g., missing else statements) while our suggestions are aimed at improving the plan quality and teaching efficiency.

6. NEXT STEPS

Our main next step is to move the human-robot interaction from simulation into a shared physical space using a PR2 robot and an approximately life-size plastic and wood mockup of the relevant parts of a car. To circumvent the need for computer vision processing, we will place the user, robot and mockup inside of a Vicon motion-capture cell and attach markers to objects as needed. This will make it possible to move away from a GUI-style interaction and towards a more natural, situated interaction. If the GUI does not totally disappear in this context, it will be moved to a handheld touch screen.

For example, instead of pressing a primitive task button, selecting the input object(s) from a menu, and then pressing the Execute button, the user will be able to physically demonstrate a primitive task by picking up, putting down, screwing, unscrewing objects, etc. With only a small number of primitive task types, we expect to easily develop a classifier to recognize which primitive action is being performed and its inputs. For learned abstract actions, a GUI (or perhaps voice recognition) may still be used to name the action, but the selection of input(s) will be done by gesture, such as pointing to or touching the appropriate object(s). We will study how these changes affect the learning process, which will then inform how other communication functions of the GUI, such as the Teach New Task and Task Complete buttons, the chat pane and the display of the current hierarchical structure should be handled.

In a more theoretical direction, our next steps include exploring how to extend our algorithms to support top-down learning and then a flexible combination of top-down and bottom-up. Among other things, this involves integrating learning from demonstration with learning from instruction techniques. We also plan to explore improvements to our existing heuristic rules and new heuristics for making useful suggestions, especially as we extend our task domain beyond tire rotation.

Finally, working with a physical robot will inevitably present us with the challenge and opportunity to explore the very broad issue of how uncertainty and failure in robotic systems affects the learning process.

7. ACKNOWLEDGMENTS

This work is supported in part by the Office of Naval Research under Grant N00014-13-1-0735.

8. REFERENCES

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[2] M. Cakmak and A. L. Thomaz. Designing robot learners that ask good questions. In *ACM/IEEE International Conference on Human-Robot Interaction*, pages 17–24. ACM, 2012.

[3] S. Chernova and M. Veloso. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34(1):1, 2009.

[4] A. Garland, K. Ryall, and C. Rich. Learning hierarchical task models by defining and refining examples. In *International Conference on Knowledge Capture*, pages 44–51, 2001.

[5] B. J. Grosz and C. L. Sidner. Attention, intentions, and the structure of discourse. *Comput. Linguist.*, 12(3):175–204, July 1986.

[6] B. Hayes. Social hierarchical learning. In *Proceedings of the 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI 2013) Pioneers Workshop*, 2013.

[7] B. Hayes and B. Scassellati. Discovering task constraints through observation and active learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[8] S. B. Huffman and J. E. Laird. Flexibly instructable agents. *Journal of Artificial Intelligence Research*, 3:271–324, 1995.

[9] S. Mohan and J. E. Laird. Towards situated, interactive, instructable agents in a cognitive architecture. In *AAAI Fall Symposium Series*, 2011.

[10] A. Mohseni-Kabir, S. Chernova, and C. Rich. Collaborative learning of hierarchical task networks from demonstration and instruction. In *RSS Workshop on Human-Robot Collaboration for Industrial Manufacturing*, Berkeley, CA, July 2014.

[11] A. Mohseni-Kabir, C. Rich, and S. Chernova. Learning partial ordering constraints from a single demonstration. In *ACM/IEEE International Conference on Human-robot interaction*, pages 248–249, 2014.

[12] M. N. Nicolescu and M. J. Mataric. Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *AAMAS*, pages 241–248, 2003.

[13] C. Rich. Building task-based user interfaces with ANSI/CEA-2018. *IEEE Computer*, 42(8):20–27, 2009.

[14] C. Rich and C. L. Sidner. Using collaborative discourse theory to partially automate dialogue tree authoring. In *Proc. Int. Conf. on Intelligent Virtual Agents*, pages 327–340, Santa Cruz, CA, Sept. 2012.

[15] P. E. Rybski, K. Yoon, J. Stolarz, and M. M. Veloso. Interactive robot task training through dialog and demonstration. In *ACM/IEEE Int. Conf. on Human-Robot Interaction*, pages 49–56, 2007.

[16] H. Veeraraghavan and M. Veloso. Learning task specific plans through sound and visually interpretable demonstrations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2599–2604, 2008.