Hierarchical Multi-Agent Reinforcement Learning through Communicative Actions for Human-Robot Collaboration

Elena Corina Grigore and Brian Scassellati
Department of Computer Science
Yale University
51 Prospect Street, New Haven, CT 06511
{elena.corina.grigore, brian.scassellati}@yale.edu

Abstract

As we expect robots to start moving from working in isolated industry settings into human populated environments, our need to develop suitable learning algorithms for the latter increases. Human-robot collaboration is a particular area that has tremendous gains from endowing a robot with such learning capabilities, focusing on robots that can work side-by-side with a human and provide supportive behaviors throughout a task executed by the human worker. In this paper, we propose a framework based on hierarchical multi-agent reinforcement learning that considers the human as an "expert" agent in the system—an agent whose actions we cannot control but whose actions, jointly with the robot's actions, impact the state of the task. Our framework aims to provide the learner (the robot) with a way of learning how to provide supportive behaviors to the expert agent (the person) during a complex task. The robot employs communicative actions to interactively learn from the expert agent at key points during the task. We use a hierarchical approach in order to integrate the communicative actions in the multi-agent reinforcement learning framework and allow for simultaneously learning the quality of performing different supportive behaviors for particular combinations of task states and expert agent actions. In this paper, we present our proposed framework, detail the motion capture system data collection we performed in order to learn about the task states and characterize the expert agent's actions, and discuss how we can apply the framework to our human-robot collaboration scenario.

1 Introduction

The field of human-robot collaboration (HRC) [1] focuses on endowing robots with capabilities to work alongside humans and help them achieve various tasks in different settings. Whether it be in public spaces, like restaurants or museums, or in our homes, robots should adapt to the task at hand and learn from the user how they can be of assistance as the task progresses. Although robotics has seen recent successes in quickly learning a couple of different manipulation tasks [2], [3], robots are still far from being able to autonomously carry out tasks requiring a high level of dexterity for different types of manipulation (e.g., using a small screwdriver on a screw that needs to pass through a nut, which itself needs to be inserted into a hole in a wooden piece). Furthermore, even beyond manipulation issues, the level of knowledge about the task necessary for a robot to autonomously build a piece of furniture, for example, is extremely high and not straightforward to acquire.

To create a useful system, we turn our attention towards semi-autonomous robots that aim to learn how to provide supportive behaviors to a person during a task, and not complete the task autonomously themselves. Such supportive behaviors are meant to help the human worker complete the task at

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

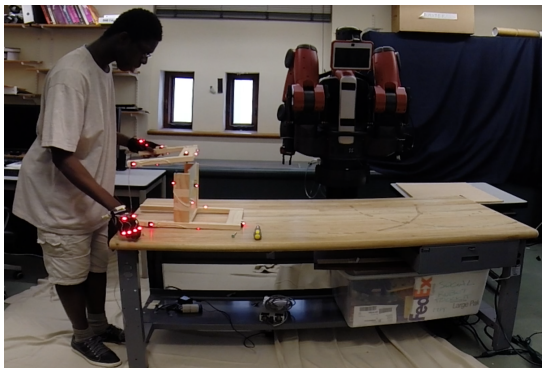

Figure 1: Participant wearing gloves fitted with sensors for motion capture system data collection.

hand more efficiently instead of breaking up the task into subtasks that the robot and user would work on individually. Although the latter would also prove useful in human-robot collaboration scenarios, we focus on complex tasks where a person could benefit from having another agent provide different types of supportive behaviors (e.g., when a person needs to use both their hands for screwing in a screw, while also needing to stabilize the piece of wood they are working on). Standard task and motion planning techniques based on configuration spaces that employ a divide-and-conquer approach cannot identify such supportive behaviors. This is because the necessary actions are not distinct components of the task given to the system a priori, but need to be discovered based on each subtask and performed concurrently with the human worker's action(s).

In order to account for different agents operating within our environment, we need to consider the different types of actions performed by each, the effects their actions have on the world they operate in, and the goals they are working towards. A relevant paradigm for representing such a problem is multi-agent systems (MAS) [4], where several agents attempt to solve tasks together in order to reach specific goals, either shared or competing. A particularly useful method that has been widely applied to MAS is that of reinforcement learning (RL) [5]. In RL, the agent operates in an environment in which it receives rewards when performing different actions in each state of the world and subsequently aims to behave such that it maximizes the long-term reward received. The use of RL for MAS has engendered the multi-agent reinforcement learning (MARL) paradigm [6], where multiple learners can take different actions in the same environment, each trying to maximize long-term reward (such agents can be collaborative, competitive, or somewhere in between). This setup, however, assumes that we can control all the agents present in the system, and that each has a clear set of defined actions it can take in the shared world. Our HRC scenario, however, involves a human whose actions we neither control, nor define. Yet, these actions are important since, not only do they cause the state of the task to change over time, but they indirectly influence and are influenced by any supportive behaviors provided by the robot, and need to be taken into account.

We thus employ a variant of the MARL framework, in which we know the state of the world (i.e. the state of the task), as well as the actions each of the agents are taking (i.e. we know what the robot's actions are, and we know what the human worker is doing during each state of the task), but we can only directly control the actions of the robot. Given that we consider the human to be knowledgeable about the task, we consider them an "expert" agent, which knows how to act within the task and has common sense knowledge about how different supportive behaviors could be useful within the task. The actions of the expert agent may be indirectly influenced by those performed by the robot, but this indirect impact is accounted for through the world dynamics (i.e. the joint actions performed by the person and the robot both affect state transitions). We use a hierarchical reinforcement learning (HRL) framework [7] in order to re-use different types of supportive behaviors for different subtasks within the same domain, as well as to help with abstracting away from the state of the subtask details about the implementation of the specific supportive behaviors.

A particularly important component of any RL algorithm is the reward the agent receives whenever it takes an action. The reward embeds domain knowledge and needs to be chosen carefully since it has great impact on the policies the agent learns. We propose the use of an interactive scheme for our hierarchical MARL framework, where the robot takes communicative actions at key points during the task in order to ask the person what supportive behavior (if any), hence what action, it should perform in that state. These key points prompting the robot to take a communicative action are based

on objective, task-related metrics (e.g., mistakes or backtracking actions the person makes indicating low effectiveness or possible need of assistance). The reward function for the robot is composed of these task-related metrics, as well as the user's response to the communicative actions. The response can tell the robot whether or not to employ a supportive behavior, and what this behavior should be in that state. This has the potential to speed up learning, as the robot need not try out all different types of supportive behaviors available at the level of those subtasks.

The rest of this paper positions our work, presents the proposed framework, details motion capture data acquired during part of an assembly task for an IKEA chair (shown in Figure 1 that depicts a participant completing the task), and discusses the implications of the proposed framework for HRC.

2 Related Work

The field of MAS is relevant to a wide array of areas, including robotics, telecommunications, distributed control, economics, and so on. A considerable amount of research studies the use of RL for MAS, the intersection of which is known as multi-agent reinforcement learning (MARL). MARL presents important benefits, such as faster learning through experience sharing (e.g., exchange of local information, skilled agents teaching less skilled ones, etc.), speed-ups due to parallel computation when exploiting the decentralized structure of the task, robustness to failures of individual agents, and easy insertion of new agents into the system [6], [4]. However, it is also plagued by a number of challenges, including the curse of dimensionality (the state action-space grows exponentially with the number of agents), the difficulty in specifying goals in such multi-agent settings (the rewards different agents receive are correlated and thus cannot always be maximized independently), non-stationarity due to concurrent learning (for each agent, the best policy changes as other agents' policies change), and an exacerbated exploration-exploitation tradeoff (agents need to explore not only to reach new regions of the state space, but also to gather information about other agents) [6], [4].

Even though there exist considerable challenges in the field of MARL, this framework is extremely relevant to our human-robot collaboration scenario. This is due to the fact that, even though it is not part of our goal to find policies for the human worker during the task, their actions affect the transition model of the world in which our robot needs to operate. Beyond directly impacting the world dynamics, the actions performed by the human also play a key role in the type of supportive behavior that is useful for the robot to offer during different subtasks. In order to address two of the main challenges mentioned above—the curse of dimensionality, and partial observability with respect to the policies of other agents—researchers have looked into the use of hierarchical reinforcement learning (HRL). Work in this area includes the use of options [8], HAMs [9], and MAXQ [10]. Other work looks at using task-level coordination in HRL settings, where all agents were given a hierarchical break down of the task at hand that was used as a basis for communication [11], [12].

Work that addressed partial observability and the issue of how the actions of other agents impact the policy of one agent includes a learning paradigm called team-partitioned, opaque-transition reinforcement learning (TPOT-RL). TPOT-RL considers opaque transitions to encompass multi-agent environments where each learner cannot rely on having knowledge of future state transitions after acting in the world [13]. Other relevant work in this area uses behaviors, defined as goal-driven control laws that achieve and/or maintain particular goals, to abstract away low level details of robot control and diminish the learning space, as well as tackling the credit assignment problem via heterogeneous reinforcement functions and progress estimators [14].

The related work presented above considered the use of MARL for scenarios where multiple robots or agents need to collaborate with each other in order to reach a common goal. We now mention research performed in the area of HRC, specifically. Such work includes enabling a robot to improve its policy for a knot untying task through user-provided guidance. In this scenario, the robot performs a set of state-action transitions to shake the contents placed inside of a bag (with the goal of untying the knot and emptying the contents of the bag), and asks for a reward signal from the human at the end of each learning episode [15]. Other relevant work in the area of obtaining rewards from a human user explored the importance of understanding the human-teacher/robot learner system as a whole, with findings that people use the reward signal not only to provide feedback about past actions, but also to provide future directed rewards to guide subsequent actions [16]. Yet other work investigated discovering policies for improving collaborator performance through a task and motion planning approach [17], which differs from our MARL framework.

Our proposed approach distinguishes itself from previous work by considering a unique view of the hierarchical MARL paradigm, where *one of the agents* is an expert and thus *does not learn* its policy simultaneously with the robot learner, but *performs actions that have a direct effect* on the dynamics of the world. We thus model this scenario as a special case of a MARL problem, where the robot needs to learn policies that adapt not only to the state of the task, but to the uncontrolled actions of the person as well. The nature of this problem is also inherently different from the above work due to the fact that we aim to learn supportive behaviors, different from the types of actions the human worker performs, and different from the types of actions a robot would learn for performing a task autonomously. To this end, we also take advantage of the interaction with a human, who can use common sense knowledge about how different types of supportive behaviors might be useful as the task progresses, and include this in the reward signal via employing *communicative actions*.

3 Existing Paradigms in Reinforcement Learning

Given that our proposed framework lies at the intersection of existing paradigms in RL, we define these paradigms by going through the formalisms used for each.

3.1 Single-Agent RL

A typical RL formulation relies on the Markov Decision Process (MDP) formalism. An MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{I} \rangle$, where $\mathcal{S}$ represents the state space the agent operates in, $\mathcal{A}$ represents the action space the agent acts within, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a probabilistic transition function (representing the dynamics of the world) such that $P(s'|s, a)$ denotes the probability of transitioning to state s' when the agent takes action a in state s, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ represents the reward function with $r(s, a)$ being the reward received by the agent when it takes action a in state s, and $\mathcal{I} : \mathcal{S} \rightarrow [0, 1]$ constitutes the initial state distribution. The typical MDP formulation for infinite-horizon settings (i.e. where the agent can take an infinite number of steps) uses discounted sum of rewards $\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$, where $\gamma \in [0; 1)$ is the discount factor. [1]

The MDP formulation is very well suited for the RL paradigm, where we have an agent that operates in a world composed of the states defined above, chooses one of several actions available to it in each state, receives a reward, and then moves into a different state based on the transition probability. The agent seeks to maximize its long-term discounted reward and thus seeks an optimal way to behave. The goal is then to solve the MDP by finding an optimal policy, π^* (a mapping $\pi : \mathcal{S} \rightarrow \mathcal{A}$ from states to actions in the deterministic case, or a probability distribution $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ over state-action pairs in the stochastic case), representing a policy that, for all states, is at least as "good" as any other policy (i.e. has a value or action-value function [2] as high as that of any other policy).

There exist several ways of solving MDPs, but here we mention Q-learning [18], an off-policy algorithm (i.e. the agent does not need to follow the policy for which it is learning the action-value function) that is also model-free, meaning that the agent does not need to learn the transition probabilities. We believe a model-free algorithm would work well in our HRC scenario since the world dynamics are difficult to estimate in a complex task (such as assembling an IKEA chair), where the person is unconstrained in the actions they can take. Model-free algorithms are temporal difference (TD) techniques [5], which allow the agent to update the estimate of the value or action-value function based on other estimates, without needing to wait for the true value.

3.2 HRL

HRL is typically formulated based on a semi-Markov Decision Process (SMDP), an extension of the MDP in which actions can take a variable amount of time to complete [19], [20]. An SMDP is defined in a similar manner, as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{I} \rangle$. The only two elements that are different from those in an MDP are the transition probability and reward functions. The former is now defined as $\mathcal{P} : \mathcal{S} \times \mathbb{N} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, where $P(s', N|s, a)$ is the probability of transitioning from state s to state s' in N time steps when the agent takes action a (the moments when transitions occur are

[1] In our assembly task, we consider an indefinite-horizon episodic setting, where the agent(s) can take arbitrarily many steps but the episode eventually ends. We can thus include $\gamma = 1$ for a finite number of steps T.

[2] The value of a state under a policy is the expected discounted sum of future rewards starting from that state and following that policy thereafter. The action-value is similar, but also considers the action taken in that state.

called decision epochs). The latter is now defined as $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, with $r(s, a)$ representing the expected total reward between two decision epochs, with the agent being in state s at the first epoch and taking action a (unlike in the MDP case, here rewards can accrue over the course of one action).

HRL is a framework for scaling RL problems with large state spaces by re-using policies learned as individual actions [7]. Such abstraction allows a learning system to ignore details irrelevant to each level of the task, and reduces computational costs. Most of the algorithms used for solving MDPs can be extended to solve SMDPs, including Q-learning. HRL generalizes the idea of a "macro-operator"—a sequence of operators or actions that can be invoked by name just like a primitive action—to closed-loop partial policies, defined for subsets of the state set [7]. An important paradigm proposed and studied in the HRL community is the formulation of partial policies into options [8].

Options generalize primitive actions to include temporally-extended courses of action [8]. An option is defined as a tuple $\langle \mathcal{I}_{\mathcal{O}}, \pi, \beta \rangle$, where $\mathcal{I}_{\mathcal{O}}$ is the initiation set of the option (the option is available in a state s if and only if $s \in \mathcal{I}_{\mathcal{O}}$), $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ is the policy to be executed, and $\beta : \mathcal{S}^{+} \rightarrow [0, 1]$ is the termination condition. When the agent takes the option, it chooses actions according to the policy π until the option terminates according to β. Our proposed framework uses options as temporally-extended options available at different levels of the hierarchy for the chair assembly task.

3.3 Hierarchical MARL

The SMDP model has been further extended to the multi-agent domain, resulting in the multi-agent SMDP (MSMDP) model. We define the MSMDP similarly to [21], as a tuple $\langle \Upsilon, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{I}, \mathcal{T} \rangle$. Υ is a finite set of n agents. Each agent $j \in \Upsilon$ has at its disposal a set of actions, A^j, together forming the joint-action space $\mathcal{A} = \prod_{j=1}^{n} A^j$. An element of $\mathcal{A}$ represents the simultaneous execution of actions a^j by all agents j and is denoted $\vec{a} = \langle a^1, a^2, ..., a^n \rangle$. Elements $\mathcal{S}$, $\mathcal{P}$, $\mathcal{R}$, and $\mathcal{I}$ are defined like in the SMDP case, with the note that $P(s', N|s, \vec{a})$ is the probability that the system transitions from state s to state s' in N steps when the agents take joint-action $\vec{a}$. Finally, $\mathcal{T}$ represents the termination scheme that specifies how decision epochs are defined, to handle the fact that temporally-extended actions can have variable lengths.

Different types of termination strategies can be classified into synchronous schemes, where all agents make a decision at each epoch (agents need to be synchronized in a centralized way), and asynchronous schemes, where only a subset of agents make decisions at each epoch (de-centralized decision making). Examples of three termination strategies, τ_{any}, τ_{all}, $\tau_{continue}$, have been researched in [22]. τ_{any} has all agents interrupt their actions as soon as the first action of the joint-action that has been executing terminates, at which point the agents choose a new joint-action. τ_{all} involves waiting until all the actions part of the joint-action terminate for the agents to choose the next joint-action, with agents who need to wait until completion taking an "idle" action. These two strategies are examples of synchronous schemes. Finally, $\tau_{continue}$ is an example of an asynchronous scheme, and involves choosing the decision epoch based on the earliest terminating action in a joint-action (like τ_{any}) but having only those agents who have terminated their actions choose new ones.

4 Expert Agent Hierarchical MARL with Communicative Actions

In this section, we describe the HRC task we are considering, and propose a paradigm for allowing the robot to learn supportive behaviors by interacting with the expert agent during this task.

4.1 Task Definition

We consider the complex task of assembling part of an IKEA chair. We use a children's chair for ease of tracking and moving, but choose one that requires complex assembly, including several small components and the need for a tool. To run several trials with each participant, we consider part of the assembly task (attaching the front frame to the rest of the chair). This part contains similar steps to those required for the whole assembly, and includes three types of small objects (pegs, nuts, and screws), as well as a screwdriver (see Figure 2). Participants need to place both pegs in either the front frame (referred to as the frame herein) or the rest of the chair (referred to as the main component herein), place both nuts in the sides of the main component (peg and nut placement can be done in any order), attach the frame onto the main component, place the screws into the two holes of the frame that are aligned with the holes and nuts in the main component, and screw them in.

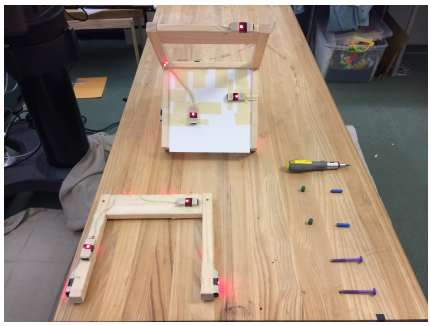

Figure 2: The components part of the assembly task. The task includes a frame, a main component, two pegs (blue), two nuts (green), two screws (purple), and a screwdriver (yellow and gray).

4.2 Proposed Paradigm

The paradigm we are proposing for the robot to learn supportive behaviors throughout our HRC task is a hierarchical MARL framework with two agents, in which we consider one of them (the person) to be an expert. The expert agent knows how to behave towards the completion of the task, as well as has knowledge of what kinds of supportive behaviors would be useful. Even though we cannot directly control the actions of the expert agent, they still influence the state of the task—hence, the state of the world—as well as what types of supportive behaviors the robot should offer, and so we consider the person an agent operating in the same environment. Furthermore, the actions the expert agent takes are influenced indirectly by the supportive behaviors the robot employs throughout the progression of the task, and so considering a MARL paradigm is important. Since we are tackling a real-world, complex task, we propose a hierarchical approach that has the advantage of utilizing state abstraction at various levels of the task. This abstraction allows us to learn policies for different subtasks, which we can re-use when we encounter similar subtasks in the future.

Another distinguishing element of our proposed framework is that of interacting with our expert agent at key points during the task in order to obtain feedback about what types of supportive behaviors to apply for different subtasks and different actions the expert agent is taking. We integrate this feedback as part of our reward function, alongside task-related metrics. This helps reduce the amount of exploration the robot needs to perform in order to learn what supportive actions to perform in different states by essentially updating the action-value of a state-action pair for multiple supportive behaviors in one step. We describe how this can be accomplished below.

Our framework is based on an MSMDP, defined as a tuple $\langle \Upsilon, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{I}, \mathcal{T} \rangle$, with components defined as follows. We consider two agents, and so the first element is the set $\Upsilon = \{r, h\}$, where r is the robot and h is the human. Of course, this model can be extended to include multiple r agents (learners) and multiple h agents (expert agents).

In order to define $\mathcal{S}$, we consider the states of the world to be the different states of the task throughout its progression. Since we are considering a complex, real-world assembly task, the states are not trivially defined. During our data collection phase, detailed in Section 5, we used a motion capture system to record the motions of users' hands during different trials of the assembly, as well as the location of the chair components in 3D space. We consider the states to be intuitively defined subtasks corresponding to nodes one would find in a task tree (e.g., a task tree based on an instructions manual including nodes such as placing a peg, or attaching the front frame). We are thus using the motion capture data set we collected to segment our task into states corresponding to subtasks from a given task tree. We have first applied the BP-HMM algorithm [23], a method that can find a global set of "primitive behaviors" present in a set of time series, and can then specify which behavior of this set is active at each time step for each time series. We applied this algorithm separately for three sets of data representing the chair components, the right hand of the human, and the left hand of the human, respectively. For detecting the states of the task, we use the data representing the chair components. In order to make sure we obtain appropriate states, we are currently manually labeling the videos by using the points where there is a change in primitive behaviors to delineate different subtasks.

The joint-action space is defined as $\mathcal{A} = A^r \times A^h$. Agent r has at its disposal a set of actions $A^r = \{a_1^r, a_2^r, a_3^r, a_4^r, a_5^r, a_6^r, a_7^r\}$, where the actions are each defined as options and represent none (no action), communicative action, stabilize object, bring object to person, take object from person,

move object closer to person on table, move object away from person on table, respectively. These options are defined at an intuitive level and are based on identified types of supportive behaviors in HRC [17]. We can, of course, extend this set as needed. Agent h performs actions we cannot control, but which are observable to us since we utilize a motion capture system that tracks the human's hand movements continuously. Using the same points found through the process described above for identifying the task states, we consider the two time series corresponding to the person's right and left hands, respectively, to represent the action the human is taking during that state. In order to discretize the person's actions, we cluster the time series and represent the expert agent's actions as $A^h = \{a_1^h, a_2^h, ..., a_{n_h}^h\}$, where n_h is the total number of resulting clusters. Alternatively, we can consider a hybrid approach, where we consider the actions of the human to be continuous and use a function over time steps that encodes the time series. An element of $\mathcal{A}$ represents the simultaneous execution of actions by both agents and is denoted $\vec{a} = \langle a^r, a^h \rangle$.

We define our reward function to have three different components, namely an exploration incentive, a task-related metric, and a social fitness component. We can write the reward as $r(s, a) = r_{none} + r_{task} + r_{social\_fitness}$. r_{none} constitutes a small negative reward for not taking any action, encouraging exploration. r_{task} represents the task-related metric, that we define as a small negative reward incurred whenever the human performs a backtracking action. In the case of our task, this represents placing a screw back on the table or removing it from the frame. This insight is based on the data we collected, having observed that people performed this backtracking type of action whenever they were not able to align the screw with the nut that needs to be placed inside of a whole in the main component of the chair. In any RL paradigm, the reward needs to encode domain knowledge, and so the task-related metric depends heavily on the task at hand. Finally, $r_{social\_fitness}$ has a value of 0 whenever the robot does not take any communicative actions, a negative reward when it takes a communicative action by asking the expert agent if help is needed and receives a "no" answer, and a positive reward when it takes the communicative action and receives a "yes" answer. When the robot incurs sufficient negative reward from the task-related metric (this would happen at key moments during the task, such as when the person is trying to attach the front frame onto the main component), the robot would eventually learn that it can take communicative actions to potentially gain positive reward. The reward function strives to balance not overloading the person with questions during every state of the task with efficiently exploring the state space.

Further, we use the communicative actions not only as part of the reward, but also to leverage the proposed hierarchical approach. Whenever the expert agent answers "yes" to a communicative action, the robot presents a list of different possible behaviors it can perform. The person can choose at least one and at most all, causing the robot to learn that for that particular state-action pair, it should try out the particular supportive behavior(s) specified by the expert agent. This has the potential of greatly reducing the need for exploring all the possible different combinations of state-action pairs. When this occurs, the hierarchical approach allows us to learn a higher-level action comprising both the communicative action and the supportive behavior that we know has the potential to be useful in that particular state-action combination. We note here that the human's response contributes to actual learning, and that the behaviors cannot just be baked into the policy. This is because we employ a task-based metric as part of the reward that greatly impacts the quality of taking different actions in different states. The human thus provides guidance to help eliminate the need to try out all the possible behaviors in every state. Given that we also use a social fitness component as part of our reward, the human's answer can also be regarded as a preference. When the person's feedback results in lower task-based performance, this trade-off should be examined. A large gap between user preference and task-based metrics might indicate that the latter should be reconsidered, or that the task contains enough variation that learning on a per-user basis is the preferable option. In our case, we consider that the robot aggregates its learning across users so that it is can help a new user.

Finally, we define the remaining components of the MSMDP, $\mathcal{P}$, $\mathcal{T}$, and $\mathcal{I}$. First, given our collected data set, we could learn the dynamics of the world after defining the task states and the human's actions. However, we are proposing using a model-free algorithm, such as Q-learning (or a new model-free algorithm for the proposed framework), in order to avoid learning the transition probabilities of such a complex task. Second, since we can only control the actions of agent r, we use a termination scheme that is similar in nature to $\tau_{continue}$, defined above. Whenever the robot finishes its action, a new decision epoch occurs, and r takes into account the action agent h is performing in order to choose its next action. Third, the set of initial states consists of a few similar states in which the different chair components are placed in a position similar to that shown in Figure 2.

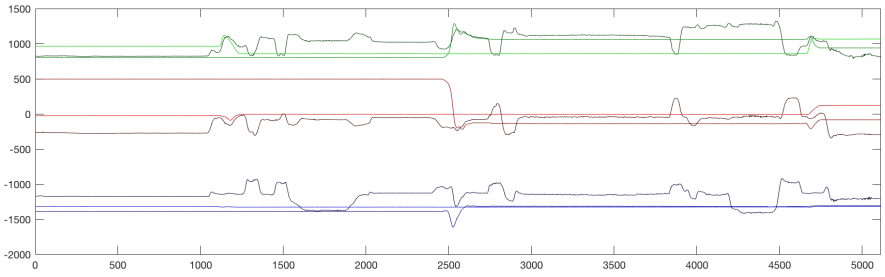

Figure 3: An example time sequence of the raw motion capture data. The dark, medium, and light shades represent the signals for the right hand, the frame, and the main component of the chair, respectively (with green, red, and blue corresponding to the x, y, and z dimensions, respectively).

5 Data Collected

We collected a data set consisting of a total of 28 participants performing the assembly task described in Subsection 4.1. Each participant performed between five and ten rounds, each consisting of both the assembly and the disassembly of the front frame. For the purposes of the proposed paradigm, we only consider the assembly recordings. We recorded each assembly as a different trial, with five-to-ten trials per participant, for a total of 158 trials across the data set. Trials varied in length, from $0:57$ minutes to $4:21$ minutes, with an average length of $2:10$ minutes. The motion capture system recorded approximately 80 samples per minute.

The motion capture system we employed throughout our data recording sessions is PhaseSpace [24], which utilizes active LED markers as its tracking technology. To acquire this data, we hand fashioned a pair of gloves specifically for participants to be able to manipulate all the different objects during this task. We used fitted, stretch fabric gloves without the tips of the fingers, which we mounted with eight sensors each for tracking. Figure 1 shows a participant wearing the gloves during one of the trials. We also tracked the frame and the main component. Figure 2 helps to show that we created two rigid bodies, one for each of the frame and the big component. We used redundant sensors placed on different planes, in order to always track the objects, even when occlusions occurred due to participants placing them on the table in different positions or covering some of the sensors.

During each trial, we recorded motion capture data for each of the 16 glove markers (eight on each glove) and all the rigid body markers. We considered position information—(x, y, z) coordinates—for the glove markers and the centers of the two rigid bodies. For each of the gloves, we averaged over the eight values corresponding to the eight markers each was fitted with, resulting in an average (x, y, z) signal for each hand. We did so based on the fact that the information from the different markers on the hand was redundant and encoded the same type of changes in position, only slightly shifted. This provided us with a good representation for the signals for each hand. Figure 3 shows an example of the data acquired during an assembly trial.

6 Conclusions

We have proposed a framework for tackling HRC scenarios in which we aim to have a robot learn to provide supportive behaviors to a person during a task. Our proposed paradigm uses a hierarchical MARL approach in which we consider the human to be an expert agent, whose actions we cannot control, but whose actions, together with the robot's actions, affect the state of the task. An important component of the framework is that of taking communicative actions in order to ask the human whether a supportive behavior is desired, and if so, which one(s). The response to the former is included as part of a social fitness component in the reward function. The response to the latter allows for the learning of higher-level actions encompassing both the communicative action and the suggested supportive behavior as good options to explore for particular state-action combinations.

We have also presented a data collection phase consisting of acquiring motion capture data from which we can extract state space discretization, as well as discretization for the expert agent's actions. An evaluation of our framework includes training the system with the proposed reward and observing whether useful supportive behaviors are provided that help increase the task-related metric (in our scenario, by decreasing the number of screw place-downs) or help reduce task completion times.

References

[1] A. Bauer, D. Wollherr, and M. Buss, "Human–robot collaboration: a survey," *International Journal of Humanoid Robotics*, vol. 5, no. 01, pp. 47–66, 2008.

[2] O. Kroemer, C. Daniel, G. Neumann, H. Van Hoof, and J. Peters, "Towards learning hierarchical skills for multi-phase manipulation tasks," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1503–1510.

[3] J. Fu, S. Levine, and P. Abbeel, "One-shot learning of manipulation skills with online dynamics adaptation and neural network priors," in *Proceedings of the 29th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016.

[4] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous agents and multi-agent systems*, vol. 11, no. 3, pp. 387–434, 2005.

[5] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[6] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," in *Innovations in Multi-Agent Systems and Applications-1*. Springer, 2010, pp. 183–221.

[7] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete Event Dynamic Systems*, vol. 13, no. 1-2, pp. 41–77, 2003.

[8] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1, pp. 181–211, 1999.

[9] R. E. Parr, "Hierarchical control and learning for markov decision processes," Ph.D. dissertation, Citeseer, 1998.

[10] T. G. Dietterich, "Hierarchical reinforcement learning with the maxq value function decomposition," *J. Artif. Intell. Res.(JAIR)*, vol. 13, pp. 227–303, 2000.

[11] M. Ghavamzadeh, S. Mahadevan, and R. Makar, "Hierarchical multi-agent reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 13, no. 2, pp. 197–229, 2006.

[12] R. Makar, S. Mahadevan, and M. Ghavamzadeh, "Hierarchical multi-agent reinforcement learning," in *Proceedings of the fifth international conference on Autonomous agents*. ACM, 2001, pp. 246–253.

[13] P. Stone and M. Veloso, "Team-partitioned, opaque-transition reinforcement learning," in *Proceedings of the third annual conference on Autonomous Agents*. ACM, 1999, pp. 206–212.

[14] M. J. Matarić, "Reinforcement learning in the multi-robot domain," in *Robot colonies*. Springer, 1997, pp. 73–83.

[15] U. Kartoun, H. Stern, and Y. Edan, "A human-robot collaborative reinforcement learning algorithm," *Journal of Intelligent & Robotic Systems*, vol. 60, no. 2, pp. 217–239, 2010.

[16] A. L. Thomaz and C. Breazeal, "Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance," in *AAAI*, vol. 6, 2006, pp. 1000–1005.

[17] B. Hayes, "Supportive behaviors for human-robot teaming." Ph.D. dissertation, Yale University, 2015.

[18] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[19] R. A. Howard, *Semi-Markov and decision processes*. Wiley, 1971.

[20] M. L. Puterman, "Markov decision processes: Discrete dynamic stochastic programming," *New York, NY: John Wiley. doi*, vol. 10, p. 9780470316887, 1994.

[21] M. Ghavamzadeh, "Hierarchical reinforcement learning in continuous state and multi-agent environments," Ph.D. dissertation, University of Massachusetts Amherst, 2005.

[22] K. Rohanimanesh and S. Mahadevan, "Learning to take concurrent actions," in *Advances in neural information processing systems*, 2002, pp. 1619–1626.

[23] M. C. Hughes, E. Fox, and E. B. Sudderth, "Effective split-merge monte carlo methods for nonparametric models of sequential data," in *Advances in Neural Information Processing Systems*, 2012, pp. 1295–1303.

[24] PhaseSpace, "Phasespace motion capture," 2013, available at http://www.phasespace.com/.