Journal of Artificial Intelligence Research 59 (2017) 495-541 Submitted 11/16; published 07/17

Perturbation Training for Human-Robot Teams

Ramya Ramakrishnan RAMYARAM@MIT.EDU
Massachusetts Institute of Technology
77 Massachusetts Ave, Cambridge, MA 02139

Chongjie Zhang CHONGJIE@TSINGHUA.EDU.CN
Tsinghua University
30 Shuangqing Rd, Haidian Qu, Beijing Shi, China

Julie Shah JULIE_A_SHAH@CSAIL.MIT.EDU
Massachusetts Institute of Technology
77 Massachusetts Ave, Cambridge, MA 02139

Abstract

In this work, we design and evaluate a computational learning model that enables a human-robot team to co-develop joint strategies for performing novel tasks that require coordination. The joint strategies are learned through "perturbation training," a human team-training strategy that requires team members to practice variations of a given task to help their team generalize to new variants of that task. We formally define the problem of human-robot perturbation training and develop and evaluate the first end-to-end framework for such training, which incorporates a multi-agent transfer learning algorithm, human-robot co-learning framework and communication protocol. Our transfer learning algorithm, Adaptive Perturbation Training (AdaPT), is a hybrid of transfer and reinforcement learning techniques that learns quickly and robustly for new task variants. We empirically validate the benefits of AdaPT through comparison to other hybrid reinforcement and transfer learning techniques aimed at transferring knowledge from multiple source tasks to a single target task.

We also demonstrate that AdaPT's rapid learning supports live interaction between a person and a robot, during which the human-robot team trains to achieve a high level of performance for new task variants. We augment AdaPT with a co-learning framework and a computational bi-directional communication protocol so that the robot can co-train with a person during live interaction. Results from large-scale human subject experiments (n=48) indicate that AdaPT enables an agent to learn in a manner compatible with a human's own learning process, and that a robot undergoing perturbation training with a human results in a high level of team performance. Finally, we demonstrate that human-robot training using AdaPT in a simulation environment produces effective performance for a team incorporating an embodied robot partner.

1. Introduction

People often require training in order to perform tasks well, particularly in areas such as disaster response, military and manufacturing. As tasks become increasingly complex and require cooperation within a team of individuals, training helps team members learn to coordinate their roles effectively and perform well under novel task variants (Gorman, Cooke, & Amazeen, 2010). Today, some team-based tasks are performed only by humans, with others performed by teams consisting solely of robots. In this work, we envision that new types of human-robot training procedures would enable robots and humans to accomplish tasks more efficiently together than they do separately.

Designing an effective human-robot training procedure poses a number of challenges: First, the method for training must support humans while learning complex team tasks, ideally leveraging existing training strategies used for human teams. The method must also be formalized for robot partners, such that computational models and algorithms for robot participation in the training process can be designed and compared with one another through empirical evaluation. The training technique must be designed for heterogeneous teams in which humans and robots have differing capabilities. Finally, both the training methodology and the robot's models and algorithms for learning must promote high levels of team performance when contending with novel tasks and environments.

Three training approaches have been widely studied in human teams: procedural, cross and perturbation training (Gorman et al., 2010). The most widely employed approach is procedural training, in which a team practices a procedure repeatedly in order to become proficient at performing a particular task. This approach results in high-level performance of the trained task, but does not generalize to tasks involving unexpected variations or disturbances, which almost always occur in real-life scenarios.

The cross training approach is designed to improve team adaptivity through practice by requiring team members to switch roles with one another. Through this process, teammates are able to build a shared mental model of the task and collaboration strategy, and can thus better anticipate one another's actions. This training method has been shown to benefit human-robot teams, but does not scale as teams increase in size and diversity (Nikolaidis & Shah, 2013). If the roles required to accomplish a task differ widely from one another, or if the given team is of a particularly large size, learning the roles of other team members becomes difficult and impractical (Gorman et al., 2010). Individual performance can also decline in such cases, as members must train for a greater number of roles.

The third approach, perturbation training, addresses the limitations of both procedural and cross training (Gorman et al., 2010), and is the human team training strategy that we build upon in this work. In this method, a team experiences slight disturbances (perturbations) during the training process that are intended to help team members learn to coordinate effectively under new variants of the given task. This training method is well-suited to heterogeneous teams, as each member practices his or her own role on the team during the training process, and it also does not require switching roles among team members, as cross-training does. Results from recent studies (Gorman et al., 2010) indicate that perturbation training yields high levels of human-team performance on novel tasks; however, this training strategy has not yet been formalized as a computational problem. As a result, there is not yet a clear pathway for incorporating an intelligent robot with algorithms for learning and communication to participate in this process alongside a human team member.

Our contributions in this work are five-fold. First, we formally define the problem of perturbation training as a variant of a multi-agent transfer learning problem involving six components: (1) multiple source tasks to use for team training, (2) one target task to use in order to test team performance, (3) a co-learning framework for incorporating human interactions into the robot's learning process, (4) a communication protocol in which human and robot partners explicitly exchange information during the action selection process, (5) an algorithm for robot learning in the source tasks, and (6) an algorithm for the robot to transfer its prior learning to the target task.

Our second contribution is Adaptive Perturbation Training (AdaPT), a novel transfer learning algorithm that enables a human-robot team to efficiently and robustly co-learn a joint strategy through training on perturbations. There are two key challenges to developing such an algorithm: First, it must rapidly learn in the source tasks to support real-time interaction with a human. Second, the transfer process must allow the robot to draw from a library of previous experiences in a manner compatible with the process of a human partner learning through perturbations. This second challenge is fundamental to the success of perturbation training, as evidenced by prior studies of human teamwork in which human teams with members possessing accurate but dissimilar mental models performed a given task more poorly than teams with members possessing less-accurate but similar mental models (Marks, Sabella, Burke, & Zaccaro, 2002). The AdaPT algorithm is a hybrid of transfer and reinforcement learning techniques, and modifies the Policy Reuse in Q-Learning algorithm (PRQL) (Fernández, García, & Veloso, 2010) to hasten learning for new task variants. In this work, we demonstrate through empirical evaluation within simulated domains that AdaPT achieves high-level performance and is faster and less sensitive to specification of priors and other parameters than other techniques for multi-agent transfer from multiple source tasks (Fernández et al., 2010; Ammar, Eaton, Taylor, Mocanu, Driessens, Weiss, & Tuyls, 2014).

In this paper, we also provide the first end-to-end computational framework for perturbation training that enables a robot to co-train with a person during live interactions. The framework includes AdaPT, as well as a co-learning method through which the human interacts with the robot during designated intermittent sessions. These sessions allow the human to gain experience by jointly performing the task with the robot, and facilitate interaction using a bidirectional communication protocol. Prior literature indicates that perturbation training scales well for training of large teams. In this work, we focus on first demonstrating the training approach for a team consisting of one human and one robot. Demonstration of the benefits of perturbation training for larger human-robot teams is left to future work.

Our fourth contribution through this work is to conduct and report on human subject experiments wherein 36 teams of two (one person and one robot) co-trained under perturbations in a simulated environment. Our results demonstrate that the framework achieved higher levels of team performance on novel tasks than those achieved through a comparable computational model for human-robot procedural training. Specifically, we found that perturbation-trained teams achieved statistically significantly higher reward than procedural-trained teams when tested on the novel task variants most different from the most recently trained task ($p < 0.05$). This result provides the first support for the idea that the relative benefits from human team perturbation training can also be realized for human-robot teams. Further, our formalization of the problem statement and computational framework provide a pathway for future study into alternate learning, transfer and interaction algorithms that improve the ability of human-robot teams to effectively learn complex joint-action tasks through experience of perturbations.

Finally, we demonstrate in robot experiments with a PR2 (n=12 human-robot teams) that human-robot perturbation training within a simulated environment results in effective performance by a team including an autonomous, embodied robot partner that communicates and receives commands through speech. This is an encouraging result, as it indicates that simulation can provide an effective and scalable approach to human-robot team training, just as simulation training typically yields high performance among human teams (Sandahl, Gustafsson, Wallin, Meurling, Øvretveit, Brommels, & Hansson, 2013; Griswold, Ponnuru, Nishisaki, Szyld, Davenport, Deutsch, & Nadkarni, 2012; Nikolaidis & Shah, 2013).

2. Related Work

We first present an overview of previous literature addressing technical approaches related to our computational model for perturbation training. Our model includes a human-robot co-learning framework, an algorithm for robot learning in the source and target tasks, and a communication protocol.

2.1 Learning with Humans

A substantial number of prior works have focused on training robots to perform complex tasks given sets of human demonstrations. In a work by Akgun, Cakmak, Yoo, and Thomaz (2012), humans physically guided a robot through a sparse set of consecutive keyframes, which were connected to perform a skill. However, this required several demonstrations of the task, which can be time-consuming for a human to provide. Alexandrova et al. aimed to reduce the number of necessary demonstrations by allowing humans to perform a task once and then provide additional information through an interactive visualization tool (Alexandrova, Cakmak, Hsiao, & Takayama, 2012). In order to take better advantage of each human teaching instance, Chernova and Veloso (2010) developed a framework to simultaneously teach multiple robots. Further leveraging demonstration data from humans, Niekum et al. proposed a technique that automatically detected repeated structures within the data by segmenting motions and constructing a finite state automaton that represents the task (Niekum, Osentoski, Konidaris, Chitta, Marthi, & Barto, 2014). This method resulted in high-level task knowledge and a set of reusable, flexible skills (such as assembling one leg of a table). These approaches support natural human interaction and help robots to learn effectively based on a small number of human demonstrations, and further enable robots to accommodate variations in the motion- or action-level execution of complex tasks.

While Learning from Demonstration (LfD) supports robot learning, it is limited in that it requires demonstrations of the given task. Other works have incorporated learning through human input by using feedback or reinforcement to improve robot decision-making. For example, the TAMER framework, developed by Knox and Stone (2009), modeled a human's reinforcement function, which a robot then used to select the actions likely to result in the most positive human reinforcement. In a work by Griffith et al. (2013), rather than using human feedback to shape rewards or Q-value functions, human inputs served directly as policy labels to infer what the human considered to be the optimal policy. In another work, Chao and Thomaz (2012) developed a system for turn-taking interactions between a human and a robot in order to collaboratively solve the Towers of Hanoi problem.

Rybski et al. (2007) presented an interactive robot task training framework in which the robot observed a human performing a task, interrupted when more clarification was needed and incorporated feedback to improve task execution.

Comparably fewer works have addressed interdependent learning between a human and a robot. Hawkins et al. (2014) developed a representation of structured tasks that the robot used to infer current human actions, as well as the execution and timing of future actions. The authors showed that a robot was able to robustly infer human actions and work effectively on a collaborative assembly task. In a work similar to our own, Oudah et al. (2015) incorporated "cheap talk" – non-binding, costless communication – to teach a robot to better learn to play competitive games through practice with a human partner. (Examples of "cheap talk" include: "I've changed my mind" or "That was not fair!") This work, however, does not address co-learning for interdependent collaborative tasks.

More closely related to our work in human-robot collaborative team training, Nikolaidis and Shah (2013) computationally modeled cross training from human team studies using a Markov decision process (MDP) model. In this approach, a human and robot switched roles during training to learn a shared model of the given task, helping the team to co-develop a convergent plan for collaborative task execution. However, this work considered a simple task with 27 states, and the approach is limited in its ability to scale to tasks involving larger state spaces or requiring complex coordination strategies.

2.2 Reinforcement and Transfer Learning

Many techniques use MDPs to represent stochastic tasks in which the agent learns how to act robustly through reinforcement learning (RL) (Busoniu, Babuska, & De Schutter, 2008; Puterman, 2009; Sutton & Barto, 1998). An MDP is defined as a tuple $< S, A, T, R >$, where S is the discrete set of world states, A is the discrete set of all possible actions, $T(s, a, s')$ is the state transition function that specifies the probability of progressing to next state s' given the current state s and action a, and $R(s, a, s')$ is the reward function that specifies the reward gained from taking action a in state s and transitioning to next state s'. The policy $\pi(s)$ learned from training specifies the action the agent should take at each state s.

In the RL framework, an agent interacts with an environment, observing state s at each time step and deciding whether or not to take action a. The environment returns a reward r and a next state s', and the agent continues taking actions until it reaches a specified goal state. The agent repeatedly executes these episodes, each of which progresses from an initial state to a goal state, in order to learn how to perform the given task well.

Q-learning (Watkins & Dayan, 1992) is an RL algorithm that learns a Q-value function $Q(s, a)$, which specifies the expected utility of taking action a from state s. The agent's goal is to learn a deterministic Markov policy $\pi(s)$, which maps each state s to an action the agent should take. The policy can be computed using $Q(s, a)$ by taking the action a with the highest value at each state s. A policy is identified as optimal if it obtains the maximum expected discounted future reward from any initial state (Sutton & Barto, 1998). While RL is one approach to learning a policy in an MDP, it can be slow if used to learn each new task separately and from scratch. Even with human input and guidance, it can be inefficient when computing policies for new task variants.

In order to hasten the learning process, transfer learning is applied in MDPs (Pan & Yang, 2010; Taylor & Stone, 2009; Torrey & Shavlik, 2009) to learn a new task given similar, previously learned tasks. Prior works have primarily considered the transfer of knowledge from one source task to one target task. Taylor et al. aimed to transfer value functions across tasks with different state and action spaces by specifying a behavior transfer function $\rho(\pi)$ (Taylor, Stone, & Liu, 2005). They created this function for the RoboCup soccer keepaway domain and showed across three different function approximators that the training time on the target task was reduced compared with learning from scratch. Mann and Choe proposed an approach to transferring knowledge from one source task to one target task that transfers action-values and acts during the new task with directed exploration (Mann & Choe, 2012). They found that it is possible to achieve positive transfer and decrease the sample complexity of exploration in the new task, even with partially incorrect mappings.

In another work, Konidaris and Barto (2007) generalized low-level, primitive actions to create temporally extended, high-level skills, called "options". For example, the option *openDoor* may consist of a series of policies that instruct the robot how to first reach, then grasp, and finally turn a door handle. To port the options over to several problems, the authors used two different representations: one in *problem-space*, which is Markovian and specific to a particular task, and another in *agent-space*, which can be non-Markovian and shared among many problems. Options are learned in agent-space rather than problem-space, rendering them reusable for other tasks with the same agent-space. In one example, the "lightworld domain," an agent uses light sensors to find a key and unlock a door. The light sensor readings are in agent-space because they are transferable across lightworld instances; however, the problem-space contains more specific information, including the room numbers, x and y coordinates of the agent and whether or not the agent currently possesses the key. Transfer learning has also been applied to model-based approaches to improve performance and sample efficiency (Taylor, Jong, & Stone, 2008). These works, however, only transferred knowledge from one source task to one target task, and cannot be directly used in situations involving multiple source tasks.

A related line of work focuses on multi-task learning (MTL), which is very similar to transfer learning. However, methods incorporating MTL assume that all MDPs are drawn from the same distribution, while transfer learning approaches do not include this assumption. Brunskill and Li (2013) developed an MTL technique that involves two phases of learning: the first performs single-task learning on multiple tasks and clusters them, and the second uses this knowledge to identify the model for the new task and to hasten learning. This approach achieved much better sample complexity compared with prior methods. In another work, Wilson et al. (2007) presented a hierarchical approach for multi-task learning. They represented the distribution of MDPs with a hierarchical Bayesian infinite mixture model; the distribution was continuously updated and served as a prior for rapid learning in new environments. However, as Wilson et al. mentioned in their study, their algorithm can be time-consuming and is not computationally efficient.

For perturbation training, we focus on the transfer learning problem without making any assumption about the distribution over MDPs. The goal is to draw from multiple source tasks, each a slight variation of one another, to assist in learning a new but related task. This requires determining which previous tasks are most relevant to the current one, and then using both that knowledge and prior experience to hasten the learning process necessary for completing the new task.

The Policy Reuse in Q-learning (PRQL) algorithm (Fernández et al., 2010; Taylor, Suay, & Chernova, 2011) is one transfer learning approach that closely aligns with the goals of perturbation training. This algorithm uses previously learned policies to intelligently explore actions in a new task. The approach is parsimonious, in that it considers the similarity of a new policy to existing policies and adds it to the library of policies only if it is sufficiently different from those already present. This algorithm, however, only addresses single-agent tasks.

In addition to policy reuse, many other mechanisms exist for selecting the most relevant source task(s) for learning a target task. In one work, Taylor, Kuhlmann, and Stone (2007) created a hierarchy of tasks ordered according to solution difficulty. This relative ordering was used to select source tasks that could be learned more quickly than the given target task, thereby reducing total training time. In another approach, Talvitie and Singh (2007) developed an expert algorithm to choose which policy to use for a new task. According to this method, the agent has a set of candidate policies, or "experts," from a previously learned MDP, and the goal is to incorporate these experts intelligently in order to perform the new task in a manner comparable to the performance provided by the best expert. In a recent work, Parisotto et al. (2016) developed another approach based on expert guidance using deep reinforcement learning. This method involved training one neural network that learns how to act in a set of source tasks through the use of experts, and using this single network to quickly generalize to new tasks. Eaton and desJardins (2006) incorporated many classifiers, each focusing on a different resolution of the data, into their work. Low-resolution classifiers can be transferred between tasks, while high-resolution classifiers are more specific to the particular task: for example, at low resolution, the task of recognizing the letter "C" can be transferred to recognize "G," but when displayed at a higher resolution, these letters have unique characteristics that differentiate them.

Many prior works have aimed to identify mappings between source and target tasks. In one work, Fachantidis et al. (2015) developed an autonomous method to probabilistically select inter-task mappings and transfer $< s, a, r, s' >$ experience tuples to the target task. To perform this transfer, the authors used two probabilistic measures from multi-task transfer learning: compliance, which helps to determine the most similar source tasks; and relevance, which helps to choose which instances to transfer from the source tasks. Another approach by Ammar et al. (2015) involved the learning of inter-task mappings for transfer using unsupervised manifold alignment in order to align the source and target task state spaces. The alignment is then used to initialize the target task policy in order to hasten learning.

Some works have incorporated explicit task similarity measures to guide the blending of multiple relevant source tasks for use in a target task. In one such work, Carroll and Seppi (2005) explored multiple measures of similarity between MDPs, including transfer time d_T, policy overlap d_P, Q-values d_Q and reward structure d_R. Within a given a set of tasks, one functioned as the target task while the others served as source tasks to speed up the computer's learning process for the target task. The measure d_T between two tasks represents the time taken to "converge" when knowledge is transferred from one task to the other. In this work, d_T is computed between the chosen target task and every other task, which requires running multiple transfer experiments. Carroll and Seppi compared the other three similarity measures to d_T in order to determine how well each measure approximates the true gain in transfer time, and reported that there was no one "best" similarity metric across the tested domains.

Eaton et al. (2008) incorporated another graph-based method in their work, in which each task was mapped onto a graph and a function learned on this graph was used to transfer parameters between tasks. Not unlike d_T in Carroll and Seppi's work, which is a similarity metric of the time taken to transfer to a new task (Carroll & Seppi, 2005), transferability in this graph is defined as the direct change in performance between learning with and without transfer. However, this method considers classification tasks solved using biased logistic regression, rather than reinforcement learning tasks. Similarly, Ruvolo and Eaton (2013) developed a transfer learning approach for supervised learning problems wherein a shared basis set of functions was learned and transferred across tasks. Recently, Ammar et al. (2014) computed an automated measure of similarity between MDPs by sampling $< s, a, s' >$, thereby comparing the similarities in transition functions. However, this method is limited in its ability to transfer the meaningful characteristics of similar execution strategies between MDPs with different transition and reward functions.

While these learning approaches are useful for transferring knowledge, not all provide a framework for multiple agents. Works involving many agents (e.g. those studied in Boutsioukis, Partalas, & Vlahavas, 2012) primarily focus on transfer from one source to a single target task. In one study, Mahmud and Ramamoorthy (2013) built upon the policy reuse concept for multiple agents to learn a policy for a new task based on policies from n previous types, and improved transfer learning efficiency through consideration of a restricted form of non-stationarity in the transition function.

For our problem, we assume that each collaborative task is performed by one human and one robot, and that there are multiple source tasks for which the relevance to the target task must be distinguished automatically. Since humans do not satisfy the form of non-stationarity discussed in Mahmud and Ramamoorthy's work (2013), we chose to augment the original PRQL algorithm (Fernández et al., 2010); however, PRQL does not address multi-agent tasks. The presence of multiple agents also introduces many challenges, one of which is communication, a key element of collaborative teams (Leonard, Graham, & Bonacum, 2004).

2.3 Multi-agent Communication

When problems involve multiple agents, these agents must communicate information about their local state and/or intentions to one another in order to ensure effective cooperation. Some works (Yen, Yin, Ioerger, Miller, Xu, & Volz, 2001; Fan & Yen, 2005) have modeled problems using Petri-Nets: in collaborative multi-agent tasks, the beliefs and states of agents can be represented using a Petri-Net, and agents can decide when to proactively "tell" and "ask" about information (Yen et al., 2001). A Petri-Net can also model multi-party conversations, which are then used through conversation pattern recognition to anticipate the information needs of other teammates (Fan & Yen, 2005). However, while these works explore multi-agent communication, they are not directly transferable to problems represented as MDPs.

Several works (Roth, Simmons, & Veloso, 2006; Xuan, Lesser, & Zilberstein, 2001; Guestrin, Lagoudakis, & Parr, 2002) have proposed communication protocols within an MDP model. In these studies, researchers selected communication actions based on what would maximize the expected joint reward. Similarly, Bowling and Veloso (2000) used opponent modeling for better action prediction within a competitive domain. Other works, such as that by Mohseni-Kabir et al. (2015), have used agent suggestions in the communication protocol and included the ability of agents to accept and reject others' suggestions. All of these works assumed a fully observable model in which information about tasks and costs associated with communication was available to other agents.

Many previous works have also investigated approaches to limited communication, wherein each agent only has local (rather than global) information about the environment. Some of these works (Goldman & Zilberstein, 2003; Williamson, Gerding, & Jennings, 2009; Wu, Zilberstein, & Chen, 2009) used decentralized, partially observable MDPs (DEC-POMDPs), where each agent takes an action and receives a local observation and a joint immediate reward. In such cases, each agent must decide when to communicate, as there is a cost associated with every communication. One of these approaches (Williamson et al., 2009) used an agent's perception of belief divergence in the team to determine whether to communicate: with a low degree of divergence, the team is coordinated and communication may not be necessary, whereas high divergence implies a need for coordination. Works using variations of POMDPs or multi-agent MDPs (MMDPs) (Shen, Lesser, & Carver, 2003; Xuan, Lesser, & Zilberstein, 2000; Roth, Simmons, & Veloso, 2005; Zhang & Lesser, 2012, 2011) also addressed limited communication with different sets of assumptions or approaches.

In other works, researchers have studied not just when agents should communicate, but also what they should communicate. In a study by Xuan et al. (2001), an agent could "tell" its local state to another agent, "query" for that other agent's local state or "sync" to exchange both local states. Roth et al. (2006) more specifically addressed what agents should communicate by determining which observations resulted in the highest expected reward when combined with the current joint belief.

Determining whom to communicate with is also important, as shown by Zhang and Lesser (2013). In their work, a coordination set was computed to determine which subset of agents to send information to, as it may be infeasible to communicate to a large number of agents. In another recent work by Amir et al. (2014), the standard MDP was augmented with an explicit representation of teammates' plans, which was updated as members shared new information with one another. Reasoning over and predicting the intentions and plans of other agents can also improve communication and coordination in multi-agent teams.

In our problem formulation, two agents (a human and a robot) must communicate with one another using knowledge encoded in the Q-value function. We drew inspiration for our approach from several works that selected communication actions based on what would maximize the expected joint reward (Roth et al., 2006; Xuan et al., 2001; Guestrin et al., 2002). Also, similar to the work by Bowling and Veloso (2000), in which opponent modeling was used to improve action prediction, we considered human actions in our model when making decisions. We store a joint Q-value function that can be used to determine the optimal joint action from every state. In the communication process, we use the Q-values and human input to decide whether, at each time step, to suggest an action or to accept or reject a human's suggestion – a method similar in spirit to the communication protocol

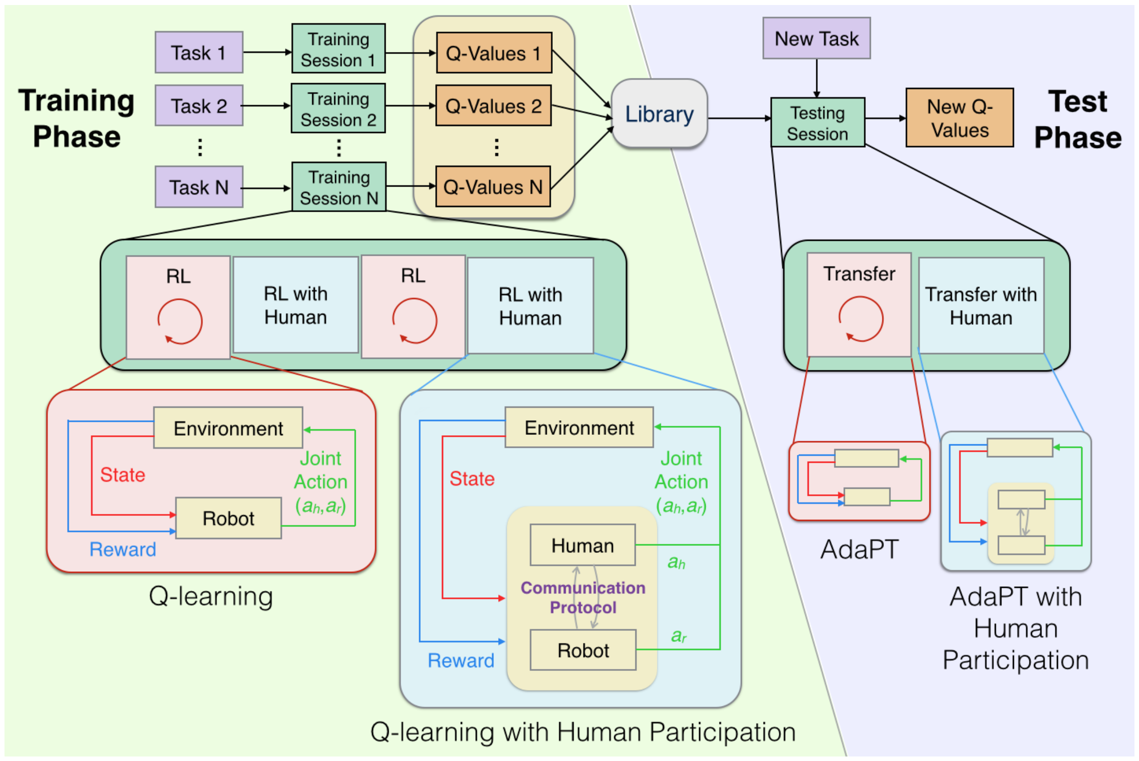

Figure 1: This figure depicts our formulation of human-robot perturbation training. In our scenario, the team experiences a training phase involving multiple source tasks and a test phase with a new target task. For each task, the robot incorporates co-learning to run multiple simulations between each human interaction. When working with a human, the robot uses a communication protocol. During the training tasks, the robot employs a simple algorithm such as Q-learning; in the test tasks, the robot uses a transfer algorithm, which in our case is AdaPT.

described by Mohseni-Kabir et al. (2015). In this paper, we assume a fully observable model and that agents communicate with one another prior to each joint action. We leave to future work an extension in which the robot must decide when, what and with whom to communicate; act under local observations; and reason over the intentions of other agents.

3. Problem Statement

We formally define human-robot perturbation training, depicted in Figure 1, as a variant of multi-agent transfer learning, consisting of the following six components: multiple source tasks, one target task, a co-learning framework, a communication protocol, a learning algorithm for the source tasks and a transfer learning algorithm.

The human and robot train on N source tasks $\{T_1, T_2, ..., T_N\}$. We require that training tasks $\{T_2, ..., T_N\}$ are variants of the original task T_1 to represent perturbations during the training process. Variants can incorporate changes to the MDP formulation, such as modifications of the transition function or reward function. After the training phase, the team receives a new task, Ω, that they have not seen before. These first two components are the same as those specified for a multi-agent transfer learning problem involving multiple source tasks and one target task.

Each task $\{T_1, T_2, ..., T_N\}$, along with Ω, is represented as a multi-agent Markov decision process (MDP) according to the tuple $\{S, A_H, A_R, T, R\}$, where:

- S is the finite set of states in the world. We assume a fully observable environment, so the state is known to all agents.
- A_H is the finite set of actions that the human can execute.
- A_R is the finite set of actions that the robot can execute.
- $T : S \times A_H \times A_R \to \Pi(S)$ is the state-transition function that specifies the probability of transitioning to state s' given the current state s, human action a_h and robot action a_r. We write the probability of the transition as $T(s, a_h, a_r, s')$.
- $R : S \times A_H \times A_R \times S' \to \mathbb{R}$ is the reward function that specifies the immediate reward for a given state s, human action a_h, robot action a_r and next state s'. We write the reward as $R(s, a_h, a_r, s')$.

Note that this two-agent formulation can be generalized to incorporate any N number of agents.

The robot learns a team policy for executing each task through a reinforcement learning process (Busoniu et al., 2008; Puterman, 2009; Sutton & Barto, 1998). The human and robot first observe the current state of the world; they then each take an action from their respective action sets (A_H or A_R). Next, they are given a new state based on what has changed in the world, as well as a feedback signal or reward. They then continue to take actions until they reach a goal state. The execution of a task, from initiation to goal state, represents one episode. The team repeats the task multiple times; through this process, the robot learns a Q-value function $Q(s, a_h, a_r)$ that specifies the value of every joint action $< a_h, a_r >$ at each state s.

The aim is to maximize the accumulated reward obtained during the task, defined as

$$\sum_{h=0}^{H} \gamma^h r_h$$

, where γ specifies the importance of future rewards and r_h specifies the immediate reward received in step h of the episode. The challenge, however, is that standard reinforcement learning techniques take many episodes to converge, and a human cannot practically be involved in every episode. In contrast to many multi-agent transfer learning systems (Kono, Murata, Kamimura, Tomita, & Suzuki, 2016; Taylor, Duparic, Galván-López, Clarke, & Cahill, 2013), we require that a human be involved in the learning process, as perturbation training for human teams requires all team members to be included. Thus, we require a framework that allows the human and robot to each gain the amount of experience necessary for high-level team performance. In this work, we define a co-learning component, where each training and testing session involves a limited number of human interactions interleaved with many robot simulation runs. This framework, depicted in green in Figure 1, allows the robot to quickly learn the task while supporting its human teammate's learning through a relatively small number of interactions. The frequency of human interactions and the number of robot simulation runs between each interaction can be modified based on the design of the co-learning framework.

Our model for perturbation training also requires a communication protocol that team members use to share information about their action selections during human interaction. For example, a protocol may specify that the human and robot are simply able to directly observe one another's actions; alternatively, another protocol may permit agents to suggest or direct actions to other teammates based on their current knowledge about the state of the world. In this work, we utilize a communication protocol similar to the latter.

The final two components of our perturbation training model pertain to the robot's learning process: The robot requires a method for learning a value function for each of the source tasks. In our work, we use the standard Q-learning algorithm (Watkins, 1989), with the MDP representation of the current task T_i as input, and a value function $Q_i : S \times A_H \times A_R \to \mathbb{R}$, as output. Any alternate to Q-learning can be used as long as the output is a learned value function $Q_i(s, a_h, a_r)$ for each source task. The robot also requires a technique for transferring the knowledge learned from practicing the source tasks in order to learn quickly during the target task Ω. Inputs to the transfer problem include the new task Ω and a library of previously learned Q-value functions $L = \{Q_1, ..., Q_N\}$ from the source tasks. The goal is to quickly learn a value function $Q_\Omega : S \times A_H \times A_R \to \mathbb{R}$ for the target task.

These six components – multiple source tasks, one target task, a co-learning framework, a communication protocol, a learning algorithm for the source tasks and a transfer learning algorithm – make up the essential elements of the human-robot perturbation training model. Our contributions in this work include the development and evaluation of a co-learning framework, communication protocol and new transfer algorithm (AdaPT). We demonstrate through results from simulation and human subject experiments that AdaPT enables a human-robot team to learn efficiently and robustly through experience of perturbations. We also show that the full model, including the co-learning framework and communication protocol, supports a human in effectively learning complex joint strategies and preserves the relevant benefits for perturbation training as compared with an alternate, commonly used human team-training technique.

4. Adaptive Perturbation Training (AdaPT) Algorithm

The goal of AdaPT is to draw from many source tasks – each a slight variation of the other – to assist in learning a new but related task. This requires determining which previous tasks are most relevant to the current one, and using both that knowledge and prior experience to hasten the learning process for the new task. AdaPT is a generalization of the Policy Reuse in Q-learning (PRQL) algorithm (Fernández et al., 2010; Taylor et al., 2011), which incorporates a library of previously learned policies to intelligently guide the search for a new policy. In PRQL, the agent explores actions that resulted in a high level of reward during previous tasks. Each policy has an associated weight that is updated during learning to reflect how well that policy performs for the new task; the higher the weight, the more likely a given policy will be chosen for the current episode. Also, as the agent learns, its confidence in these weights increases, as represented by a temperature parameter associated with an annealing process. Initially, past policies heavily guide exploration; however, after multiple iterations, the agent relies solely on the newly learned policy for execution of its new task.

The AdaPT algorithm augments PRQL to learn more quickly for new task variants. While PRQL can more intelligently guide exploration, it does not take full advantage of prior knowledge to learn well for new tasks. PRQL begins with a value function initialized naively with zeroes and uses previously learned policies to guide the learning of this new function. In our algorithm, we instead directly adapt previously learned knowledge to learn more quickly for a new task. To do this, AdaPT incorporates Q-value functions rather than policies from previously learned tasks in order to adapt knowledge more easily. With every simulated execution of the new task, the algorithm updates the values in all value functions to reflect what the agent has learned, allowing it to simultaneously adapt all value functions to be more helpful for the new task. Finally, AdaPT returns the value function with the highest weight as the new policy. This is distinguished from PRQL, which always returns the newly learned policy. In both AdaPT and PRQL, if this new Q-value function is sufficiently different from those that already exist in the library, it can be inserted as a new function, allowing the agent to continuously build knowledge. In this work, we do not test the continuous learning process of building a knowledge base, but focus instead on applying and evaluating the efficacy of our algorithm as part of a perturbation training framework. We leave analysis of AdaPT in a lifelong learning system to future work.

Figure 2 depicts the difference between AdaPT and PRQL. Here, the Q-value functions are represented as points in space; two value functions that are close to one another indicate that they achieve similar reward on a particular task than two value functions that are further apart. Q_1, Q_2 and Q_3 are three value functions learned from three training tasks. Q^*_{new} represents the unknown optimal value function the agent must learn for the new task. PRQL starts with an initial value function Q^{init}_n and uses the three other value functions to guide this new function closer to Q^*_{new}. The three functions Q_1, Q_2 and Q_3, however, are unchanged. AdaPT instead adapts all value functions from training to move them closer to Q^*_{new}. Finally, the algorithm chooses the one closest to Q^*_{new}. To approximate which value function is closest, AdaPT chooses the one with the maximum weight, as this represents the value function that has resulted in the highest reward during the new task over time. This new value function can then be added to the library and used for future tasks. Our method of directly adapting prior knowledge to learn in new situations was inspired by the way in which humans adapt: by matching new tasks to previous similar cases (Klein, 1989).

Inputs to the AdaPT algorithm (Figure 3) include the new task to perform, Ω; a copy of the Q-value functions that the robot learned from practicing on n training tasks, $\{Q_1, ..., Q_N\}$; the initial temperature τ used for policy selection; the temperature decay, $\Delta\tau$; the maximum number of episodes, K; the maximum number of steps in each episode, H; and standard γ, α, and ϵ parameters used for ϵ-greedy Q-learning. Given these inputs, the robot simulates in order to learn a joint Q-value function for the new task Ω.

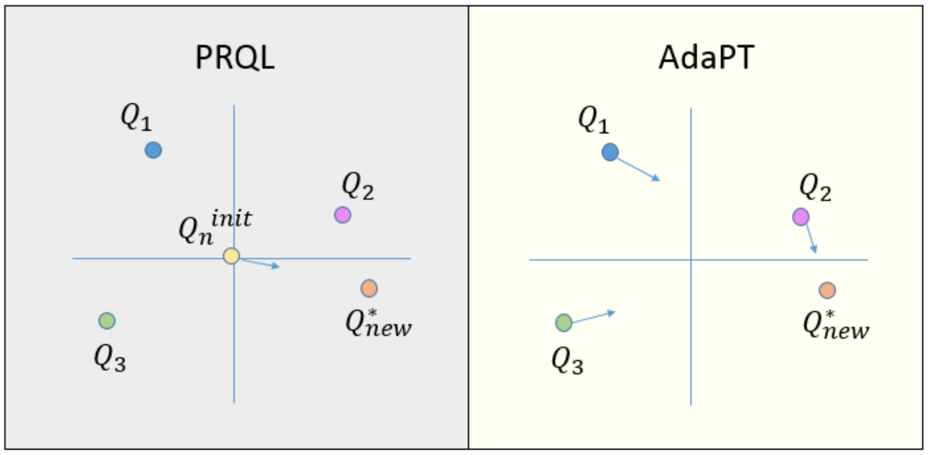

Figure 2: This chart depicts the difference between AdaPT and PRQL. While PRQL learns a new value function guided by prior knowledge, AdaPT directly adapts previously learned value functions in order to learn well during new tasks.

The algorithm begins by initializing the weights of all Q-value functions from the library to 0 in line 1. Line 2 initializes the counter for each value function, which tracks the number of episodes in which a value function is selected for execution. In Line 3, AdaPT simulates for K episodes, each time choosing a Q-value function Q_c to execute based on the current temperature τ and the weights $W_1, ..., W_N$ (line 4). The chosen Q_c is used for action selection with an ϵ-greedy approach (with probability ϵ, a random action is chosen; with probability 1-ϵ, the action with maximum value is taken). Then, all value functions are updated based on the reward received (line 5) for improved performance on the new task. Figure 5 depicts the elaborated pseudocode for this function, Update-QValues. While all value functions in the library are updated, note that only the value function chosen for the current episode, Q_c, has its weight updated in line 6. In lines 7-8, the counter for value function Q_c is incremented and the temperature parameter is updated. Lines 4-8 are repeated for K episodes; finally, in line 10, the Q-value function with maximum weight is returned for use in the new task.

The AdaPT algorithm presented in Figure 3 is similar to the main PRQL algorithm, but PRQL differs in the way it updates its Q-values: in place of Update-QValues, PRQL executes the π-reuse algorithm. Here, we present π-reuse and then our modified version, Update-QValues, and highlight important differences between them.

The π-reuse algorithm (Figure 4) takes as input the Q-value function being learned for the new task, Q_Ω; the past policy, Π_{past}, chosen to guide exploration during the current episode; the ψ parameter, which determines the probability that the past policy is chosen over an ϵ-greedy approach; the v parameter, which indicates how much to decay the value of ψ in each step; the maximum number of steps per episode, H; and standard γ, α, and ϵ parameters used in ϵ-greedy Q-learning. The algorithm returns the reward R obtained in that episode, along with an updated Q-value function Q_Ω. The episode is initialized with a randomly selected state s (line 1) and the initial ψ value (line 2). With probability

Algorithm: AdaPT (Ω, $\{Q_1, ..., Q_N\}$, τ, $\Delta\tau$, K, H, γ, α, ϵ)

```
1.  Set the weights of all Q-value functions: $W_i = 0$, $\forall i = 1, ..., N$
2.  Set the number of episodes Q-values $Q_i$ has been chosen $U_i = 0$, $\forall i = 1, ..., N$.
3.  **for** $k = 1$ **to** $K$.
4.      Choose a Q-value function for the current episode, where the probability
        of selecting a value function as $Q_c$ is determined by the following equation:
            $P(Q_c) = \frac{e^{\tau W_c}}{\sum_{p=1}^{N} e^{\tau W_p}}$ $\forall c = 1, ..., N$
5.      $< R, Q_1, ..., Q_N >$ = Update-QValues($Q_1, ..., Q_N$, $c$, $H$, $\gamma$, $\alpha$, $\epsilon$)
6.      Set $W_c = \frac{W_c U_c + R}{U_c + 1}$
7.      Set $U_c = U_c + 1$
8.      Set $\tau = \tau + \Delta\tau$
9.  **end for**
10. Return Q-value function with maximum weight
```

Figure 3: The Adaptive Perturbation Training (AdaPT) algorithm takes as input a new task and a library of value functions learned from previous tasks, among other parameters. It adapts all value functions simultaneously and returns the value function with the highest weight to execute in the new task.

ψ, the algorithm executes one step that follows the past policy action $\Pi_{past}(s)$ and, with probability 1 - ψ, executes the action determined by the new policy using an ϵ-greedy approach (lines 4-5). Reward is received (line 6), and the Q-values are updated in line 7 using the Q-learning value update. Line 8 decays the ψ parameter so that the past policy is used less frequently in later episodes. In line 9, the state is updated. Line 11 calculates the accumulated reward over this episode, discounted using γ. Finally, the reward and the updated Q-value function are returned in line 12.

Here, we present the Update-QValues algorithm (Figure 5), which takes as input the Q-value functions $Q_1, ..., Q_N$; the index of the Q-value function, c, chosen for action selection in the current episode; the maximum number of steps per episode, H; and standard γ, α and ϵ parameters used in ϵ-greedy Q-learning. The algorithm returns the reward R obtained during that episode, along with updated Q-value functions $Q_1, ..., Q_N$. The episode is initialized at a randomly selected state s (line 1). The algorithm then executes a random joint action with probability ϵ in line 4. With probability 1 - ϵ, the joint action with maximum value in the current Q-value function Q_c is chosen (line 6). The reward is received in line 7, and all Q-value functions are updated in line 8 using the Q-learning value update. In line 9, the state is updated; in line 11, the accumulated reward is calculated over this episode, discounted using γ. Finally, the reward and the updated Q-value functions are returned in line 12.

Algorithm: π-reuse (Q_Ω, Π_{past}, ψ, υ, H, γ, α, ϵ)

```
1. Set the initial state $s$ randomly
2. Set $\psi_1 \leftarrow \psi$
3. for $h = 1$ to $H$
4.     With probability $\psi$, $< a_h, a_r >= \Pi_{past}(s)$
5.     With probability 1-$\psi$, $< a_h, a_r >=$ $\epsilon$-greedy($\Pi_\Omega(s)$)
6.     Receive next state $s'$ and reward $r_h$
7.     Update: $Q_\Omega(s, a_h, a_r) = (1-\alpha)Q_\Omega(s, a_h, a_r) + \alpha[r_h + \gamma max_{a'_h, a'_r} Q_\Omega(s', a'_h, a'_r)]$
8.     Set $\psi_{h+1} \leftarrow \psi_h \upsilon$
9.     Set $s \leftarrow s'$
10. end for
11. $R = \sum_{h=0}^{H} \gamma^h r_h$
12. Return $< R, Q_\Omega >$
```

Figure 4: π-reuse is a subfunction of PRQL that executes one episode of the task (from initialization to goal state) by using the chosen past policy to guide the learning of a new value function. In AdaPT, we replace π-reuse with Update-QValues.

The main difference between PRQL, which uses π-reuse, and AdaPT, which uses Update-QValues, is that PRQL incorporates past policies to guide the learning of a new value function Q_Ω, while AdaPT, instead of learning a new value function, updates all previously learned value functions and ultimately chooses the one with the greatest weight. For action selection, PRQL chooses to use either a past policy, the new value function being learned Q_Ω or a random action based on ϵ. In contrast, AdaPT only considers a value function from the library or a random action based on ϵ, and does not learn a separate value function Q_Ω. This means that AdaPT does not require the ψ and υ parameters that PRQL uses to balance between Π_{past} and Q_Ω. Thus, AdaPT uses fewer parameters and takes greater advantage of previously learned value functions by directly adapting them rather than using them to guide the learning of a new value function.

Note that AdaPT and Update-QValues explicitly represent both the human and the robot in the state and action spaces; learning $Q(s, a_h, a_r)$ enables the robot to learn and make decisions over this joint space. Although this increases the complexity of the problem, it also allows the robot to consider human actions in its decision making and select optimal joint actions for the team.

$Q(s, a_h, a_r)$ is sufficient for offline learning in simulation, where the robot chooses actions for both itself and a simulated human. However, working with a real human poses an additional challenge, in that the robot does not make decisions on behalf of the human. Therefore, it is also useful to store and update a local value function $Q(s, a_r)$ during the learning process, as this enables the robot to consider the best action over all possible

Algorithm: Update-QValues ($Q_1, ..., Q_N$, c, H, γ, α, ϵ)

```
1.  Set the initial state $s$ randomly
2.  for $h = 1$ to $H$
3.      With probability $\epsilon$,
4.          $< a_h, a_r > =$ Random joint action
5.      With probability 1 - $\epsilon$,
6.          $< a_h, a_r >= max_{a'_h, a'_r} Q_c(s, a'_h, a'_r)$
7.      Receive next state $s'$ and reward $r_h$
8.      Update all Q-value functions $Q_i$, $\forall i = 1, ..., N$:
        $Q_i(s, a_h, a_r) = (1-\alpha)Q_i(s, a_h, a_r) + \alpha[r_h + \gamma max_{a'_h, a'_r} Q_i(s', a'_h, a'_r)]$
9.      Set $s \leftarrow s'$
10. end for
11. $R = \sum_{h=0}^{H} \gamma^h r_h$
12. Return $< R, Q_1, ..., Q_N >$
```

Figure 5: Update-QValues is a subfunction of AdaPT that executes one episode of the task (from initialization to goal state) and updates all Q-value functions to adapt to the new task.

human actions when the next step of the human is uncertain. (Note that it is also possible to store $P(a_h|s, a_r)$ and use this to estimate $Q(s, a_r)$.)

It is also important that the robot's learning algorithm run quickly during human interaction. We analyzed the time complexity of AdaPT thusly: it runs for K episodes, with each episode executed for up to H steps. At each step, all N previous Q-value functions are adjusted using the Q-learning value update. Each update requires an estimation of the optimal future value, which iterates over all $|a_h| \times |a_r|$ possible joint actions. Thus, the time complexity of AdaPT is $O(KHN|a_h||a_r|)$. Parameters such as K and H are set according to the size of the state and action space of the task.

5. Computational Results

In this section, we empirically validate that AdaPT learns quickly and robustly for new task variants through comparison with other state-of-the-art hybrid reinforcement and transfer learning techniques. Specifically, we compared the performance of AdaPT to PRQL (Fernández et al., 2010; Taylor et al., 2011); PRQL-RBDist, a variation of PRQL that uses a recent deep-learning algorithm to choose an informative prior (Ammar et al., 2014); and standard Q-learning across two domains with state space sizes of 3,125 and 10,000, respectively.

As noted previously, the standard PRQL algorithm begins with an uninformative prior, which can result in slow learning. Therefore, we manually seeded PRQL with different priors for a fair basis of comparison. Since the performance of PRQL varies substantially with the prior, we also compared AdaPT to a variant of PRQL in which the prior is selected through the use of RBDist, an automated MDP similarity metric (Ammar et al., 2014). The technique computes similarity between MDPs by sampling $< s, a, s' >$ and training RBMs.

All four algorithms were trained on the same training tasks and tested on a variety of test tasks. For each training task, the robot learned a joint Q-value function and a joint policy through Q-learning that included both human and robot actions. For AdaPT, all value functions learned through training were given as input to the algorithm, which was run for a limited number of iterations on the new test task prior to evaluation. For PRQL, we gave as input all policies learned from training and then initialized the value function by either using an uninformative prior or by selecting one of the various value functions learned from training. We evaluated the performance of PRQL using each of these priors.

For RBDist with PRQL, we implemented the similarity measure described by Ammar et al. (2014) using Tanaka and Okutomi's implementation of the RBM (Tanaka & Okutomi, 2014). We ran RBDist for each training task and the test task given data points in the form of $< s, a, s' >$ for each task MDP. The output of the algorithm was the mean reconstruction error, which is a measure of similarity between the two input MDPs. The training task that achieved the lowest mean error was identified as the training task most similar to the new task. To incorporate RBDist into the learning process, we ran RBDist using N samples from each training task and then the first N samples of the new test task, with N specified according to the domain. We then initialized the value function of the new task with the values of the most-similar training task, and continued execution in the test task using PRQL.

Here, we present how the algorithms performed during the test tasks given limited simulation time, and compare the average accumulated reward of each technique averaged over 50 simulation runs. For AdaPT and Q-learning, we ran the test tasks with only one set of parameters; thus, we simply present the average reward for each test task and do not show variance across parameters or priors. For PRQL, we present the range in performance as we changed the prior, and report that performance is highly sensitive to the prior selected. For PRQL-RBDist, we varied the number of hidden units in the RBM between 1, 5, 100 and 500, and found that the algorithm's performance varied greatly as a result. Furthermore, RBDist takes significantly longer than AdaPT to compute a measure of similarity, which can be especially detrimental when a robot is working with a person in real time. Overall, our algorithm is able to adapt more quickly to new task variants, without the sensitivities associated with selection of a prior or the number of hidden units.

5.1 Fire Extinguishing Task

The first task was a disaster response scenario that involved a team of one human and one robot extinguishing five fires associated with five neighboring houses on a street block. The goal of the task was to put out the fires as quickly as possible while minimizing damage to the houses. In variants of this task, each of the fires was initialized at differing levels of intensity, from level 0 (no fire) to level 3 (large fire). During task execution, fires could potentially increase by one intensity level up to a maximum of 4, which indicated that the house had burned to an unrecoverable state. A fire's intensity would decrease more quickly if both the human and robot extinguished it together rather than working separately on different fires.

The probability that a fire's intensity would increase to level 4 was greater if the environment had high dryness conditions, and the probability that a fire would spread to neighboring houses was greater if the environment had high wind conditions. The team had to determine the best way to work together depending on the environmental conditions. The first, "base" task the team trained on incorporated no wind or dryness ($w = 0$ and $d = 0$). The second training task, which was the first perturbation the team experienced, included some wind but no dryness ($w = 5$ and $d = 0$). The third and final training task involved no wind and some dryness ($w = 0$ and $d = 5$). The teams were then tested using combinations of wind and dryness.

Information about the state of the world, available actions, consequences and goal criteria were encoded for the robot in an MDP. This MDP was formulated as a tuple $\{S, A_H, A_R, T, R\}$, where:

- S is the finite set of states in the world. Specifically, $S =< I_A, I_B, I_C, I_D, I_E >$, $I_i \in [0, 4]$, where each value I_i in the vector denotes the intensity of fire i and is an integer ranging from [0,4]. Intensity 0 indicates no fire, 1-3 indicate low- to high-intensity fires and 4 indicates burnout. The size of the state space is 3,125.

- $A_H \in \{A, B, C, D, E, WAIT\}$. Actions A, B, C, D, E each represent the human attempting to extinguish fires A, B, C, D, E, respectively. $WAIT$ is a no-op in which the human takes no action.

- $A_R \in \{A, B, C, D, E, WAIT\}$ represents the robot's attempts to extinguish the fires.

- $T : S \times A_H \times A_R \rightarrow \Pi(S)$ specifies the transition function. For perturbed variants of the task, wind and dryness levels affected the transition function, as the probability of a fire spreading was based on both the wind level and the intensity of the fire. If the wind level w was greater than 0, the probability of a fire spreading at intensity 3 was $(w * 10 + 10)\%$, while the probability at intensity 2 was 10% less and a further 10% less at intensity 1. If the wind spread the fire to a neighboring house, the intensity of the neighboring fire increased by one level. At most, five fires increased in intensity from the fire spreading at each time step. If the dryness level d was greater than 0 and a fire was at level 3, the probability of a house burning down during the subsequent step was $(d * 10 + 10)\%$.

  T also modeled the effect of human and robot actions on the fires during all tasks. Both team members working on the same fire had a 90% probability of decreasing its intensity by three levels, and a 10% probability of decreasing it by two levels. Separately extinguishing fires resulted in a decrease of one level with 90% probability and of two levels with 10% probability.

- $R : S \times A_H \times A_R \times S' \rightarrow R$ specifies the reward function. For the experiment task, -10 reward was assigned for each time step, and an additional negative reward was

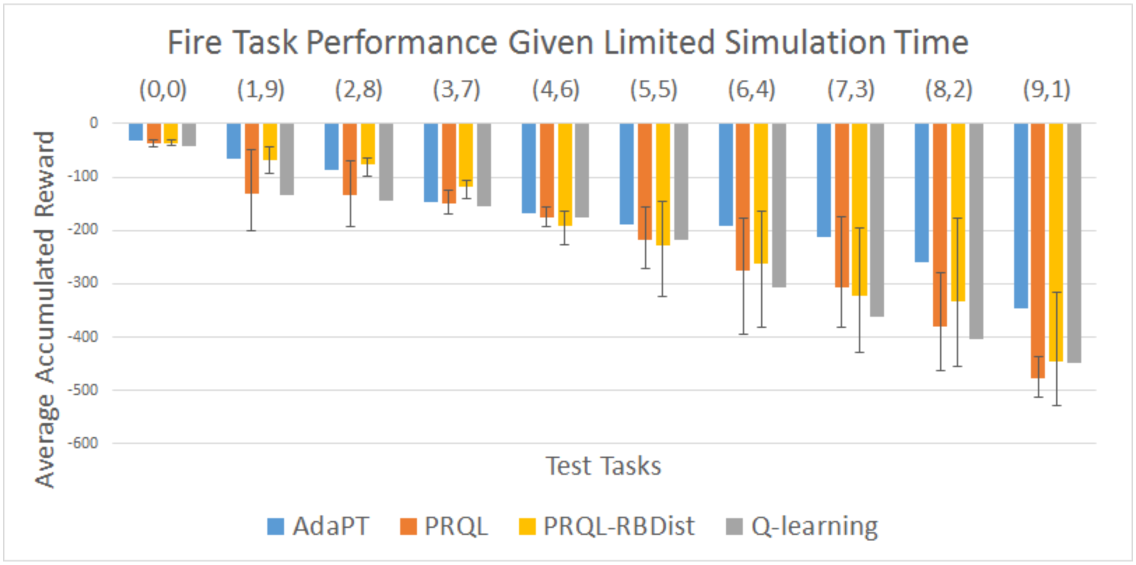

Figure 6: This chart compares the performance of AdaPT, PRQL, PRQL-RBDist and Q-learning on the fire task, given limited simulation time. The x-axis displays the wind and dryness values of the test tasks (e.g., [7,3] represents a task with wind=7 and dryness=3). The y-axis depicts the average accumulated reward received during an execution of the task after learning on the test task for 2,000 iterations. Less negative reward denotes better task performance.

added in direct proportion to the intensities of the fires at each time step. At task completion, -100 reward was assigned for each house that had burned down.

In our experiments, we used $\gamma = 1$ to represent a finite horizon problem. Other parameters were initialized as follows: $\epsilon = 0.1$, $\alpha = 0.05$, $\tau = 0$, $\Delta\tau = 0.01$, $K = 400{,}000$ for training and 2,000 for testing, and $H = 30$. For PRQL, the additional ψ parameter was initially set to 1 and the v parameter to 0.95. For PRQL-RBDist, we used 5,000 data points of the form $< s, a, s' >$ for each task.

We first compared the four algorithms: AdaPT, PRQL, PRQL using RBDist and Q-learning, given limited simulation time. We ran PRQL four times, each time initialized with either an uninformative value function or the value function learned from each of the three training tasks. We ran each of these 50 times and determined the average performance for each prior. We plotted the average accumulated reward across all priors and then depicted the range with error bars by plotting the minimum and maximum performance achieved across each of the priors. As indicated in Figure 6, PRQL's performance changed drastically depending on the prior.

For PRQL-RBDist, we ran the PRQL algorithm, using the RBDist measure as a prior selection mechanism, for a range of hidden units: 1, 5, 100, and 500, averaged over 50 runs. We plotted the average accumulated reward over all of the hidden units and presented the range in performance across them. While RBDist helped to automate the process of selecting a prior, it was also very sensitive to changes in parameters and was thus unable to achieve robust performance on the test tasks. Q-learning performed the worst on most test tasks because it does not use prior knowledge to learn new tasks more quickly. Overall, AdaPT was able to achieve better performance than the other algorithms on most tasks, without sensitivity to selection of prior or the number of hidden units. AdaPT also took far less time to execute 2,000 iterations than PRQL-RBDist: While AdaPT took 0.575 seconds on average, PRQL-RBDist took between 1.573 seconds (when using 1 hidden unit) and 77.790 seconds (when using 500 units).

5.2 GridWorld Task

The second domain was a collaborative version of a GridWorld task, similar to the evaluation domain used by Fernández et al. (2010). The objective was for a team consisting of two agents to reach the goal location while maximizing the number of tokens obtained, as in the work of Agogino and Tumer (2005), and minimizing the number of pits crossed over. If the two agents collectively obtained a token, the team received a higher reward than if they collected tokens separately. The MDP for this task was formulated as a tuple $\{S, A_H, A_R, T, R\}$, where:

- S is the finite set of states in the world. Specifically, $S =< (H_{row}, H_{col}), (R_{row}, R_{col}) >$. The human's position on the grid is indicated by (H_{row}, H_{col}), and the robot's position is indicated by (R_{row}, R_{col}). Each value denotes the row or column of the agent's location on the 10x10 grid. The size of the state space is 10,000.

- $A_H \in \{UP, DOWN, LEFT, RIGHT, WAIT\}$. These actions correspond to movement in each of the four cardinal directions on the grid. $WAIT$ is a no-op in which the human takes no action.

- $A_R \in \{UP, DOWN, LEFT, RIGHT, WAIT\}$ represents the robot's intended actions for moving around the grid.

- $T : S \times A_H \times A_R \to \Pi(S)$ specifies the transition function. The stochasticity of the task is modeled similarly to GridWorld tasks, in which the robot has an 80% chance of taking the action it chose, a 10% chance of moving to the left of that action and a 10% chance of moving to the right.

- $R : S \times A_H \times A_R \times S' \to R$ specifies the reward function. For this task, -1 reward was assigned to each time step. The human or robot earned +1 reward if they individually collected a token, but received +5 if they collected the token together, in order to encourage collaboration. If either the human or robot landed on a pit, -5 reward was given. The team received +20 reward when both the human and robot reached the goal location.

The 10x10 grid had fixed token and pit locations, as shown in Figure 7. The team trained on four training tasks, each of which involved a different goal location. In our experiments, the four corners of the grid served as the goal locations, as specified by the green cells in the figure. After training, the team received a randomly selected goal location for each test task. Parameters for the GridWorld task were initialized as follows: $\gamma = 1$, $\epsilon =$

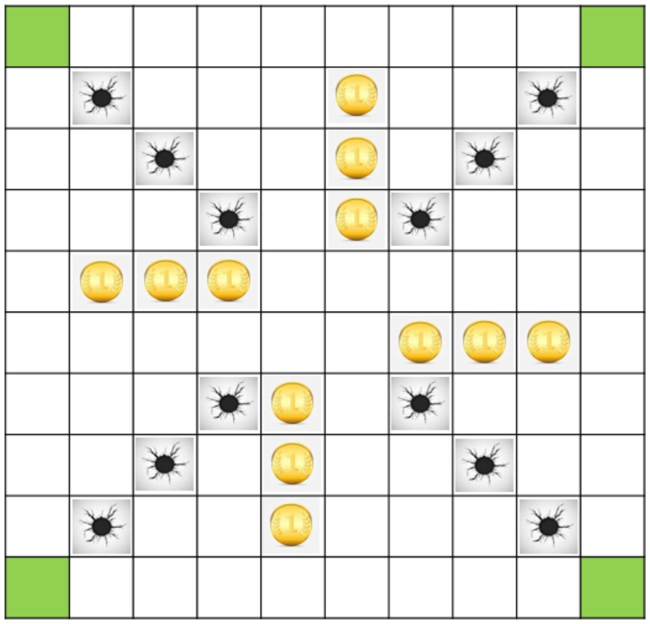

Figure 7: This figure is a representation of the 10x10 GridWorld task domain. Yellow tokens and black pits are placed in fixed locations on the grid. The four green locations represent the training task goal locations. The team was tested on 50 random test goal locations.

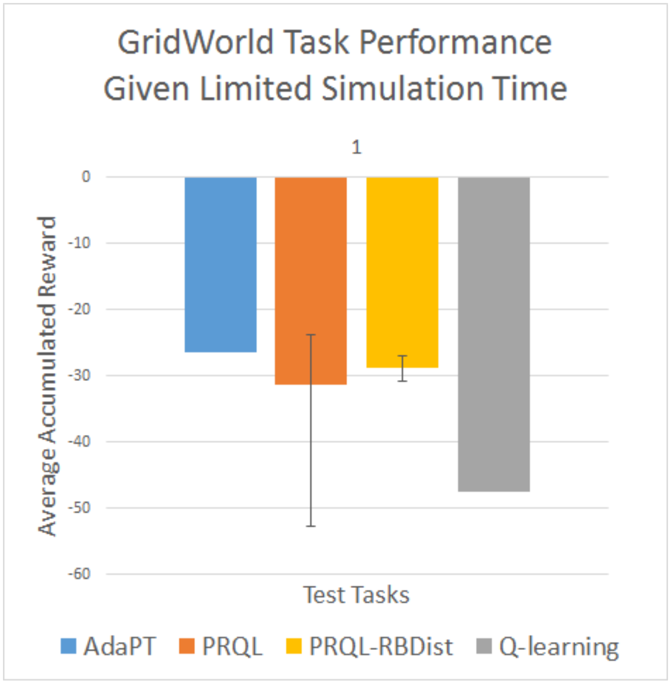

Figure 8: This chart compares the performance of AdaPT, PRQL, PRQL-RBDist and Q-learning on the GridWorld task given limited simulation time. The y-axis depicts the average accumulated reward received on an execution of the task after learning on the test task for 5,000 iterations. Less negative reward denotes better performance on the task.

0.1, and $\alpha = 0.05$, $\tau = 0$, $\Delta\tau = 0.01$, $K =$ 1,000,000 for training and 5,000 for testing, and $H = 30$. For PRQL, the additional ψ parameter was initialized to 1 and the υ parameter to 0.95. For PRQL-RBDist, we used 10,000 data points of the form $< s, a, s' >$ for each task.

As in the fire task, we compared AdaPT, PRQL, PRQL using RBDist and Q-learning given limited simulation time. We ran PRQL five times, each run initialized with either an uninformative value function (as in the original PRQL algorithm) or the value function learned from each of the four training tasks, averaged over 50 runs. We plotted the average accumulated reward across all priors and then identified the range by plotting the minimum and maximum performance of the priors as error bars. As shown in Figure 8, PRQL's performance changed drastically depending on the prior. However, this was mainly due to PRQL's negative performance when using an uninformative value function; when PRQL was initialized with any of the learned value functions, it performed much better, with a small variance across the learned priors.

A similar effect occurred in the PRQL-RBDist case. To test PRQL-RBDist, we ran the PRQL algorithm, using the RBDist measure as a prior selection mechanism, for a range of hidden units: 1, 5, 100, and 500, averaged over 50 runs. We plotted the average accumulated reward over all of the hidden units and determined the range in performance across them. While RBDist was highly sensitive to changes to the number of hidden units on the fire task, RBDist had a smaller range during the GridWorld task. We hypothesized that this is because the strategies employed were vastly different for the fire task, and choosing an inappropriate prior can significantly hinder performance. In the GridWorld task, using any prior can drastically improve learning, and choosing one value function over another does not reduce performance as heavily as it would during the fire task. Starting with an uninformative value function, however, significantly lowers performance in the GridWorld task because the state space is much larger, and starting with no learned values results in much slower learning. Finally, Q-learning performed worse than all other tested conditions because it was initialized with no prior knowledge at each test task.

Even though PRQL-RBDist was comparable to AdaPT in terms of performance during the GridWorld task, it took much longer to execute: While AdaPT took 2.259 seconds on average to execute 5,000 episodes, PRQL-RBDist took between 4.214 seconds (when using 1 hidden unit) and 17.112 seconds (when using 500 units). This indicates that AdaPT is able to execute episodes much more quickly than PRQL-RBDist, enabling the agent to learn and adapt to new tasks more quickly.

5.3 Summary of Results

These computational results indicate that AdaPT is able to achieve task performance within the two domains tested in this work in a manner comparable to or better than other evaluated methods; also, that the algorithm is fast and not sensitive to specification of priors or other parameters associated with deep learning techniques. While PRQL can perform well given the appropriate prior, it is evident from previous work that determining which training task is most similar to a new task is not trivial (Taylor, Kuhlmann, & Stone, 2008; Ammar et al., 2014). The RBDist measure provides a mechanism for prior selection, but the result is sensitive to parameter selection. It also takes a long time to train the RBM, which may be impractical for human-robot teamwork, during which decisions must be made quickly and in real time. In the next section, we show that AdaPT can provide the basis of a framework for quick adaptation that supports effective human-robot co-learning.

6. Human-Robot Perturbation Training Model

While our AdaPT algorithm provides a framework for adapting to new task variants, it does not support actual interaction with humans. Therefore, we developed a co-learning framework and a bidirectional communication protocol to better support teamwork. These two components, combined with our AdaPT algorithm, form our computational human-robot perturbation training model.

6.1 Co-learning Framework

Often, robots solve complex problems in simulation by practicing over many iterations using reinforcement learning. However, this is impractical when working with a human, as it can take hours or even days to compute optimal policies for complex problems. Furthermore, robots learning a task offline is not representative of the way humans learn collaboratively.

To better support interaction with humans, we added a co-learning framework through human-robot practice sessions. The robot simulates between each interaction with a human, allowing it to refine its policy, or “catch up” its learning, before working with that person again. During the trials between human interactions, the robot simulates experience using the approximate model and updates its value function using standard Q-learning value updates. The robot determines the optimal action for both itself and the human during these simulations, operating under the assumption that the human action is rational. This is acceptable since the primary purpose of the simulated trials is to gain more knowledge and obtain a more accurate value function. When the robot works with that individual again, this updated value function is used to select actions and communicate. After the interaction is completed, the robot again simulates over many trials and further updates its value function to incorporate this new knowledge.

6.2 Human-Robot Communication Protocol

Communication between team members is vital for effective team collaboration (Leonard et al., 2004). The Human-Robot-Communicate algorithm, depicted in Figure 9, describes how a robot communicates with a human in order to coordinate task execution. Our protocol requires that the team communicate at each step of the task (although this could be relaxed in future work in order to reduce communication overhead). At each step, the teammate who initiates communication can either suggest actions for both teammates or simply update the other teammate as to what their own action will be, allowing the teammate to make a decision accordingly. If the initiator makes a suggestion, the other teammate has the option to accept or reject it.

The algorithm utilizes two threshold values, ϵ_{sugg} and ϵ_{acc}, to determine whether or not the robot should suggest an action or accept a suggestion, respectively, at each communication step. A low threshold for suggestion ϵ_{sugg} results in the robot frequently offering suggestions that it is confident are best for the team. A high threshold means that the robot trusts the human to make correct decisions, and will only offer suggestions itself if the optimal joint action reward is much higher than the average reward across all possible human actions. A low threshold for accepting human suggestions ϵ_{acc} means that the robot will frequently reject human suggestions; too low of a threshold can be detrimental to team performance if two equally viable actions have slightly different Q-values and decisions are sensitive to these small changes. A high threshold for accepting suggestions means that the robot will accept human suggestions even if they do not yield high reward. This can be useful if the robot is attempting to act according to human preference rather than the optimal Q-value function. These thresholds may be set statically or tuned dynamically to reflect human or robot expertise in the task. We include an analysis later in this section of how human action selection and threshold values affect performance.

The way in which we incorporate communication into the learning process of AdaPT is as follows: At each time step, instead of simply calculating the optimal joint action $< a_h^*, a_r^* >$, as in line 6 in Update-QValues, the optimal joint action or human input $< a_h^*, a_r^* >$ is passed into our communication algorithm, Human-Robot-Communicate. The algorithm is then executed with human input received as necessary through text or speech, and then a final joint action is returned. Update-QValues then resumes execution by incorporating this joint action and receiving the next state. In our experiments, the human and robot alternated with respect to who initiated communication at each time step. This can be modified, like the threshold values, in future work such that a teammate initiates based on who possesses more knowledge about the task or who has important information to convey.

The Human-Robot-Communicate algorithm takes as input a tuple that specifies the human action a_h^* and the robot action a_r^* before the communication step. If the human initiates communication, these actions originate from human input; if the robot initiates, the actions originate from calculation of the optimal joint action. We input two threshold parameters: ϵ_{sugg}, to determine whether to make a suggestion or an update; and ϵ_{acc}, to determine whether to accept or reject suggestions made by the human. The joint Q-value function $Q(s, a_h, a_r)$ and the robot Q-value function $Q(s, a_r)$ are also taken as input. Using these parameters and human input, the Human-Robot-Communicate function returns the final joint action agreed upon during communication.

When the robot is initiating communication (line 1, Case 1), it has no knowledge about what the human will do, and thus has to make a decision that takes all possible human actions into consideration. It first calculates the optimal action a_r' over all possible subsequent human actions (line 2). If the joint action has a much higher value (as determined by the threshold ϵ_{sugg}) than the expected value over all human actions of $Q(s, a_h, a_r')$ (line 3), the robot suggests the optimal joint action in line 4. If the value of the joint action is not much better than the average value over all human actions, the robot simply updates the human as to its plan of action, allowing the person to choose his or her own action (line 6).

If the human initiates communication (line 7, Case 2), the inputted joint action, $< a_h^*, a_r^* >$, is either a suggested joint action provided by the human or an update from the human with a_r^* being null. The robot first calculates the optimal robot action a_r' given the human action a_h^* (line 8). If the human only communicated an update and a_r^* is null (line 9), the robot would simply return $< a_h^*, a_r' >$ in line 10. If the human made a suggestion, the robot would calculate, given a_h^*, the difference between taking the calculated optimal robot action a_r' and taking the action suggested by the human a_r^*. If the difference is greater than ϵ_{acc} (line 12), the robot rejects the suggestion and returns $< a_h^*, a_r' >$ in line 13. If the

Algorithm: Human-Robot-Communicate ($< a_h^*, a_r^* >$, ϵ_{sugg}, ϵ_{acc}, $Q(s, a_h, a_r)$, $Q(s, a_r)$)

```
1.  Case 1: Robot is the initiator
2.    Calculate the optimal $a'_r$ using $Q(s, a_r)$
        $a'_r = \arg\max_{a_r} Q(s, a_r)$
3.    if ($Q(s, a_h^*, a_r^*)$ - $E_{a_h}[Q(s, a_h, a'_r)]$) $> \epsilon_{sugg}$
4.      Return $< a_h^*, a_r^* >$
5.    else
6.      Return $< a_h, a'_r >$ where $a_h$=received human input
7.  Case 2: Human is the initiator
8.    Calculate optimal $a'_r$ given human action $a_h^*$
        $a'_r = \arg\max_{a_r} Q(s, a_h^*, a_r)$
9.    if $a_r^*$ == null (human provided an update)
10.     Return $< a_h^*, a'_r >$
11.   else (human provided a suggestion)
12.     if ($Q(s, a_h^*, a'_r)$ - $Q(s, a_h^*, a_r^*)$) $> \epsilon_{acc}$
13.       Return $< a_h^*, a'_r >$
14.     else
15.       Return $< a_h^*, a_r^* >$
```

Figure 9: The Human-Robot-Communicate algorithm provides a computational framework for the robot to make decisions when communicating with a person.

calculated optimal joint action is not much better than the suggestion, the robot accepts and returns $< a_h^*, a_r^* >$ in line 15. We use this two-way communication protocol because providing suggestions and updates is a natural way for a team to coordinate on joint tasks.

6.2.1 Robustness of Team Performance to Communication Threshold Parameters and Human Action Selection Strategy

Our communication protocol includes two threshold parameters: ϵ_{sugg} and ϵ_{acc}. We investigated the robustness of team performance as a function of the settings of these parameters, as well as with respect to human action selection strategy. We conducted this study in simulation, since conducting experiments with human subjects across such a wide range of settings would be time-prohibitive.

We considered three different types of simulated humans: a rational human, who always chooses optimal actions based on the value function; a semi-rational human, who chooses with uniform probability from the top three actions; and a human exhibiting random behavior, who chooses one of the feasible actions with uniform probability. For each human type, we tested threshold values of {0, 2, 5, 10} for both thresholds (suggestion and accept), resulting in 16 conditions. We ran each of the 16 conditions for 50 simulation runs and determined the average accumulated reward.

Our hypothesis was that, for a given human type, team performance would be robust to different settings of the communication thresholds. This is desirable because it would indicate that the robot's learning process is relatively robust with respect to the implementation of the communication protocol. However, we also hypothesized that we would observe an improvement in team performance when the communication threshold was calibrated to the human type – e.g., the robot makes fewer suggestions to a rational human and more suggestions to an irrational human. This outcome is desirable because it would indicate that dynamic tuning of the communication strategy based on the behavior of a human partner may benefit team performance.

We found that for each of the three human types, the variance in performance as a function of communication parameters was relatively small compared with the magnitude of the reward. The coefficient of variation, or the ratio of the standard deviation to the mean, was 7.4%, averaged across all test tasks. We include here the graph for the randomly behaving human (Figure 10), since this condition yielded the most variance on average. The 10 different test tasks are positioned along the x-axis, where the numbers indicate the wind and dryness within the environment (e.g., [1,9] indicates level 1 wind and level 9 dryness). The y-axis represents the average accumulated reward. For each test task, we plotted the average over all 16 threshold conditions, and the error bars indicate the standard deviation across these conditions. As depicted in the graph, the standard deviation is low relative to the overall reward obtained during each task. This is encouraging, as it supports our first hypothesis: that the settings of the thresholds would not substantially change team performance.

Next we conducted a more focused analysis of the data to determine whether calibration of thresholds according to human type confers a systematic benefit to team performance. Specifically, we conducted pairwise comparisons of team performance when thresholds were set to their extreme values: ϵ_{sugg} set to 0 and 10, indicating the robot will tend to give many or few suggestions, respectively; and ϵ_{acc} set to 0 and 10, indicating the robot will accept many or few suggestions, respectively. We also evaluated the extreme methods of human action selection: random and rational. The following tables depict the accumulated reward received on 10 test tasks in the fire domain – the same tasks used to evaluate AdaPT in Section 5. In each table, we highlight the condition that achieved the highest reward on that task. To simplify the comparisons, we indicate the threshold values by specifying who is controlling action selection: the robot has higher action control when $\epsilon_{sugg} = 0$ and $\epsilon_{acc} = 0$, and the human has higher control when $\epsilon_{sugg} = 10$ and $\epsilon_{acc} = 10$.

Table 1 depicts performance as a function of thresholds for the random human type. We expected that when the human acted randomly, it would yield better results if the robot exerted more control over action selection. Our findings indicate that when the robot dominated action selection in this manner, equal or better performance was achieved compared with human-dominated scenarios for 80% of the tasks. This supports our intuition, as allowing a randomly behaving human to dominate action selection could significantly hurt performance.

Table 2 depicts performance as a function of thresholds for the rational human. The number of times each condition resulted in the highest team performance was balanced,

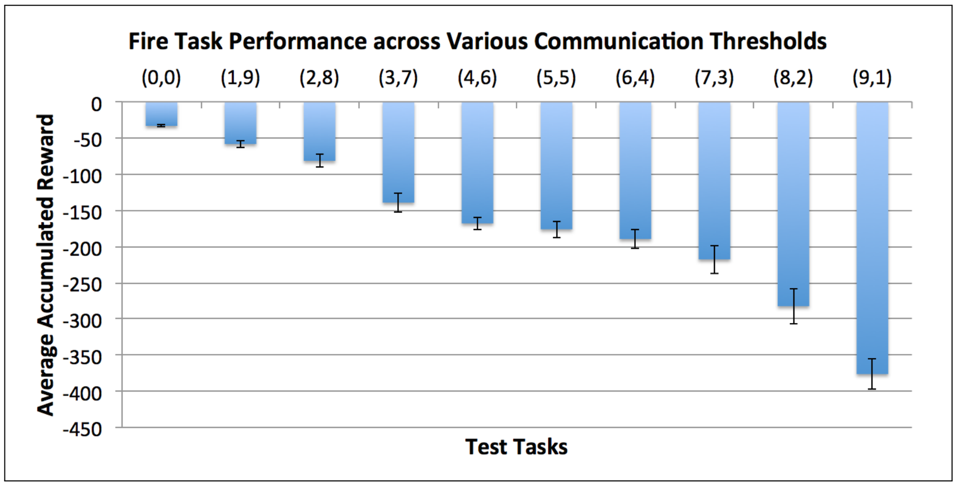

Figure 10: This chart depicts variance in team performance across multiple settings of the communication threshold parameters. The wind and dryness values of the test tasks are represented along the x-axis (e.g. [7,3] indicates a task with wind=7 and dryness=3). The y-axis represents the average accumulated reward received on an execution of the task. The error bars show the standard deviation across the 16 different settings of the thresholds.

Human Type	Action Control	1	2	3	4	5	6	7	8	9	10
Random	Robot	-32	-55	-85	-116	-158	-185	-206	-211	-258	-378
Random	Human	-34	-55	-98	-158	-162	-196	-173	-209	-289	-384

Table 1: Human Behaving Randomly

with four for the robot-dominated scenario and six for the human-dominated scenario. As expected, the threshold values did not impact performance, since both the human and robot chose the optimal actions. Under these circumstances, no matter who controls action selection, the team will perform well.

Human Type	Action Control	1	2	3	4	5	6	7	8	9	10
Rational	Robot	-33	-54	-97	-130	-179	-180	-171	-191	-298	-359
Rational	Human	-34	-46	-75	-128	-160	-176	-194	-200	-263	-396

Table 2: Rational Human

Next, as presented in Tables 3 and 4, we kept the threshold values constant and changed the human type. When the robot exerted a greater degree of control over action selection, the number of times the highest team performance was achieved under each condition was equal between the two control scenarios, at five each. This result is attributable to the fact that human type matters less when the rational robot is the dominant teammate.

Human Type	Action Control	1	2	3	4	5	6	7	8	9	10
Random	Robot	-32	-55	-85	-116	-158	-185	-206	-211	-258	-378
Rational	Robot	-33	-54	-97	-130	-179	-180	-171	-191	-298	-359

Table 3: Dominant Robot

In contrast, when the human controlled action selection, teams led by a rational human achieved equal or better performance on the majority of tasks (80%) compared with teams led by a randomly behaving human.

Human Type	Action Control	1	2	3	4	5	6	7	8	9	10
Random	Human	-34	-55	-98	-158	-162	-196	-173	-209	-289	-384
Rational	Human	-34	-46	-75	-128	-160	-176	-194	-200	-263	-396

Table 4: Dominant Human

Our analysis of human action selection and communication threshold parameters provides insight into how the communication framework can be used to change the dynamics of a team and impact performance. In aggregate, the performance of AdaPT did not change drastically with respect to these parameters. However, careful calibration of thresholds according to human type can yield meaningful improvements in team performance. In future work, the setting of the thresholds could be adapted to incorporate prior information about the relative strength of human or robot expert knowledge. Further, the parameters could be adapted during the task based on experience to prioritize the teammate who communicates more high-reward actions.

7. Human Subject Experiment Setup

We conducted human-subject experiments with both simulated and embodied robots to confirm that our human-robot perturbation training model enables a robot to draw from its library of previous experiences in a way compatible with the process of a human partner learning through perturbations.

This is a challenging task, as evidenced by previous human teamwork studies that demonstrated that human teams with members possessing accurate but dissimilar mental models performed worse than teams with members with less-accurate but similar mental models (Marks et al., 2002).

Specifically, we aimed to answer the following questions through our experiments:

1. Does our human-robot perturbation training model support a person in the process of learning through perturbations and achieving high-level team performance under new task variants?

2. Can a team achieve high-level performance by simply experiencing perturbations regardless of the learning algorithm used by the robot, or does the algorithm affect team performance significantly?
3. Does training on perturbations using AdaPT sacrifice performance on the base task compared with procedurally trained teams who train repeatedly only on the base task?
4. Can training in a simulated environment using AdaPT transfer to effective team performance with an embodied robot?

7.1 Hypotheses

To answer the first two questions, we compared two models of perturbation training: one using our AdaPT algorithm and the other incorporating the standard Q-learning algorithm with no library. Both conditions included the co-learning framework and the communication protocol. We performed this comparison in order to analyze the effect of the human's experience of perturbations compared with the effect of the robot's learning algorithm. If both algorithms achieved a high level of team performance, this would indicate that the human can adequately compensate for deficiencies in the robot's learned model by training through perturbations. However, if AdaPT performed much better than perturbation Q-learning, the algorithm that the robot used was key to achieving high-level team performance.

H1 *AdaPT teams that train on perturbations will achieve significantly better outcomes on objective and subjective measures of team performance compared with perturbation Q-learning teams on the majority of test tasks, including both novel task variants and variants similar to the training tasks.*

We hypothesized that the robot's use of a library of prior experiences would be compatible with human strategies for decision making, as numerous findings from prior studies have demonstrated that exemplar-based reasoning involving various forms of matching and prototyping of previous experiences is fundamental to our most effective decision-making strategies (Newell, Simon, et al., 1972; Cohen, Freeman, & Wolf, 1996; Klein, 1989). For example, results from naturalistic studies have indicated that skilled decision makers in the fire service incorporate recognition-primed decision making, in which new situations are matched to typical cases where certain actions are appropriate and usually successful (Klein, 1989).

To answer the third question listed above, we hypothesized that AdaPT would not sacrifice performance on the base task by training under perturbations. While human team literature indicates that procedurally trained teams that train on only one task will perform better on that task than perturbation-trained teams, we hypothesized that our model would maintain performance on the base task while also adapting to new task variants.

H2 *Teams that train on perturbations using AdaPT will achieve significantly better outcomes according to objective and subjective measures of team performance as compared with teams that train procedurally on the base task using Q-learning for novel task variants. The teams will achieve comparable measures of performance on the base task.*

Finally, to answer the fourth question, we aimed to demonstrate that human-robot training in a simulation environment using AdaPT yields effective team performance with an embodied robot partner.

H3 *Teams that train on perturbations using AdaPT in a simulation environment and then execute novel task variants with an embodied robot partner will achieve measures of performance comparable to those achieved by teams that execute the novel task variants in the simulation environment.*

This hypothesis is motivated by results from human-robot cross-training experiments that indicated people were able to work effectively with a physical robot after training in a virtual environment (Nikolaidis & Shah, 2013). The study incorporated a virtual environment that displayed a workspace and a robot, mirroring a physical task and robot. The virtual robot moved within the scene using the same controller as the physical robot. In our study, the simulation environment did not display a virtual robot; therefore, the question to be addressed is whether this result persists when the simulation environment supports the learning of strategies as sequences of discrete actions but does not support familiarization with robot motions. The question of whether non-technical skills, such as teamwork and communication, learned in a virtual environment will effectively translate to improve embodied interactions is an active area of research for human team training as well (Kim & Byun, 2011; Marshall & Flanagan, 2010).

For all hypotheses, we measured objective team performance using the reward accumulated during task execution. We also identified the number of times that participants accepted and rejected actions suggested by the robot, assuming that a larger proportion of accepted suggestions indicated that the participants agreed more frequently with the robot's decisions with regard to completing the task effectively and quickly. Finally, we measured the participants' subjective perception of team performance using a Likert-style questionnaire administered at the end of the experiment.

7.2 Experiment Methodology

Forty-eight teams of two (one human and one robot) were tasked with extinguishing a set of fires. All human participants were recruited from a university campus. The task and computational model were identical to those described in Section 5.1. Thirty-six of the teams performed both their training and testing in a simulation environment; the remaining 12 teams underwent training in simulation and testing with a physical robot. The experiment was conducted between-subjects. Each participant in the simulation-only experiment was randomly assigned to undergo either perturbation training using AdaPT, perturbation training using Q-learning or procedural training using Q-learning. All participants in the mixed simulation-hardware experiments were assigned to perturbation training using AdaPT.

Perturbation teams trained together on three different task variants, two times each. For the perturbation AdaPT condition, the robot learned a value function for each variant using Q-learning, and these Q-value functions became the library given as input to AdaPT. For the perturbation Q-learning condition, one set of Q-values was updated using Q-learning during perturbation training. This training scheme was deliberately chosen to involve a relatively small number of interactions for learning complex joint coordination strategies across multiple task variants. Each procedural team trained together on one variant of the task a total of six times. For these tasks, we used Q-learning throughout and transferred Q-values between tasks to learn one Q-value function over the six rounds. Figure 11 depicts the structure of each of these conditions.

The teams were then tested on three new variants of the task and evaluated on their performance. All teams received the same three test variants, which differed from the tasks used during training. To simulate time pressure, team members received a limit of 10 seconds in which to communicate with one another and decide their next action.

For all sessions, the participant and the robot communicated using Human-Robot-Communicate, alternating with regard to who initiated communication. The first communication during the test sessions was always initiated by the robot in order to maintain consistency. The teammate initiating communication had the option to either suggest the next actions for both teammates or to simply update the other teammate as to their own next action (i.e., which fire they planned to extinguish next).

7.2.1 Training Procedure

The experimenter introduced participants to the task using a PowerPoint presentation, which included a description of the fire scenario, the goal for the task, the human-robot communication protocol and a tutorial for the simulation interface. To give the robot equivalent initial knowledge before beginning the task, we ran Q-learning offline for 500,000 iterations over the entire state space on a deterministic version of the task without wind or dryness. Each participant then completed two practice sessions in order to become familiar with the interface. For the first practice session, participants received 15 seconds per time step to communicate with the robot to decide on an action. From the second practice session on, participants received 10 seconds per step.

The experiment GUI displayed five fires in a row, representing the five house fires. Each fire was annotated with its intensity level, with 1 indicating low intensity, 2 medium intensity, 3 high intensity, and 4 a burned-out/unrecoverable state. During task execution, the GUI prompted the participant to communicate with the robot and displayed feedback regarding its chosen actions.

After the familiarization phase, the participants underwent three training sessions. Each session consisted of two execution rounds to provide the human and robot two attempts at the same task variant, allowing the participant to learn from the first round and improve his or her coordination strategy for the second. Each round consisted of a complete episode, from the initial fire state to the goal state. Prior to the first round, the robot was provided 200,000 iterations to learn an effective joint policy for the task. Between the first and second rounds, the robot again received 200,000 iterations to refine its policy. This framework allows the robot to simulate between each human interaction and co-learn with the human. The 200,000 iterations took between 10 and 50 seconds to compute, depending on the difficulty of the task. During this interval, participants completed a short questionnaire prompting them to respond to Likert-style statements regarding their subjective perceptions of the team's and the robot's performance (similar to the questionnaire used in prior work, see Nikolaidis & Shah, 2013).

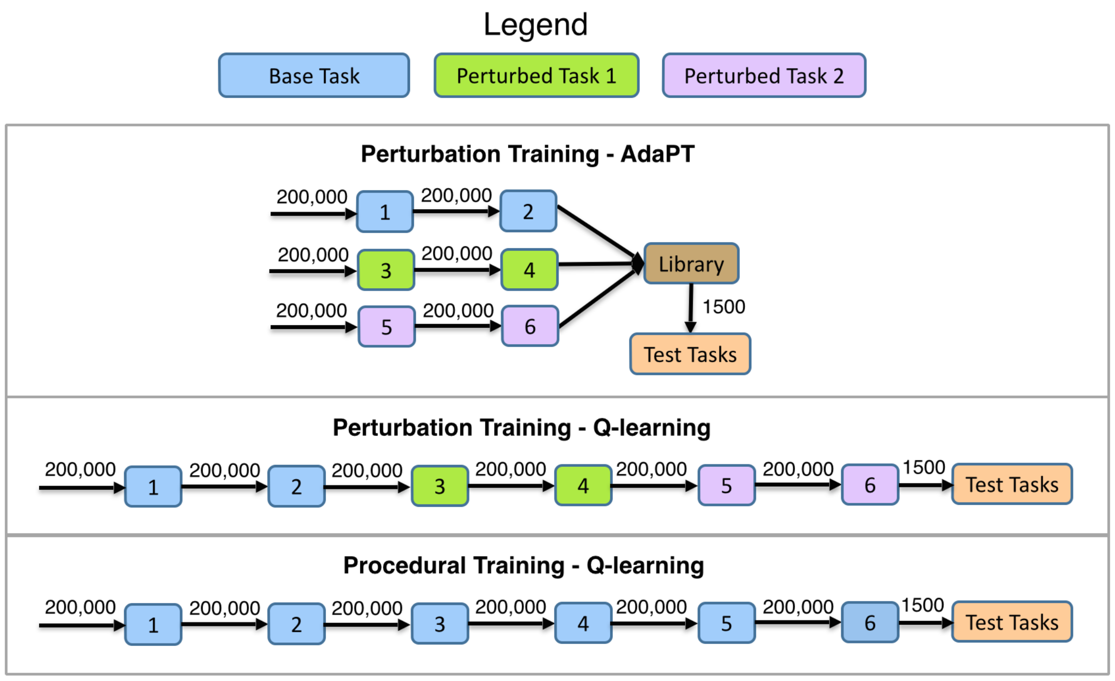
Legend
Base Task
Perturbed Task 1
Perturbed Task 2
Perturbation Training - AdaPT
200,000
1
2
3
4
5
6
Library
1500
Test Tasks
Perturbation Training - Q-learning
Procedural Training - Q-learning

Figure 11: This figure depicts the structure of the robot's learning for the conditions used during the human-subject experiments. Each box represents one round of the experiment, in which the robot executes an episode (from an initial state to a goal state) with the participant. The numbers above the arrows indicate the number of simulated episodes the robot experiences before the next round with the human. The colors indicate the variant of the task the team is executing (base task or a type of perturbation), as depicted in the legend. The perturbation AdaPT condition stores a library and uses the AdaPT algorithm. The perturbation Q-learning condition updates one value function for all perturbed tasks. In the procedural Q-learning condition, one value function is updated based on repeated experience with the base task.

In the procedurally trained teams, the human and robot trained on the same environmental conditions, $w = 0$ and $d = 0$, for all sessions. For perturbation-trained teams, the first training session was identical to the first session experienced by the procedural training group. The second and third sessions, however, included slight environmental changes that affected the progression of the fires: The second session involved some wind ($w = 6$) and no dryness ($d = 0$), allowing the fires to spread, while the third session involved no wind ($w = 0$) and some dryness ($d = 6$), resulting in more burnout from high-intensity fires. Information about wind and dryness conditions was provided to the participants and the robot, with some noise. The robot thus trained on an approximate model of the environment; this was intended to mirror real-world circumstances in which a robot will not always have an accurate model of the given task.

7.2.2 Testing Procedure

Following the training sessions, participants performed three test tasks. The first test-session task was similar to the procedural training tasks in that it involved no wind or dryness; however, the fires were initialized at different intensities than in the training tasks. The remaining two test-session tasks involved different wind and dryness levels. These task variants were more drastic than the training sessions experienced by the perturbation-trained teams and involved non-zero levels of both wind and dryness: Test task 2 had $w = 2$ and $d = 9$, and test task 3 had $w = 9$ and $d = 2$. During the test phase, the robot was constrained with regard to the time allotted for learning, and had to make a decision within a reaction time similar to that of a human (250 ms), which corresponded to 1,500 iterations.

For the simulated robot experiments, all three testing sessions were executed using the same GUI screen, with no deviation from the training protocol. For the embodied robot experiments, the participants trained in simulation and then worked with a PR2 robot during three testing sessions, which were identical to those in the simulation experiments. To represent each fire, three lights were placed underneath cups on a table and controlled using an arduino. The lights were activated to indicate the intensity of each fire: one light for low intensity, two for medium, three for high and zero lights for no fire or burnout. Additionally, the same GUI screen from the training sessions was used to display the state of the fires, the team's action choices and the time remaining for each time step. The participant and the robot communicated using a defined language structure. Google web speech recognition was used to identify human speech commands, and errors from the recognition system were manually corrected by the experimenter. A button was placed next to each of the five representations of fire for both the human and the robot. To extinguish a fire, the appropriate button was pressed by each teammate and the lights changed appropriately to reflect the subsequent state of the fire. Before the testing sessions began, participants received a brief overview of the setup and the language protocol and cooperated with the robot to put out low-intensity fires in order to practice working with the PR2. The experiment setup for the human-robot experiments is depicted in Figure 12.

8. Human Subject Experiment Results

In this section, we summarize the statistically significant results and insights obtained from the experiments. (We defined statistical significance at the $\alpha = .05$ level.) For objective

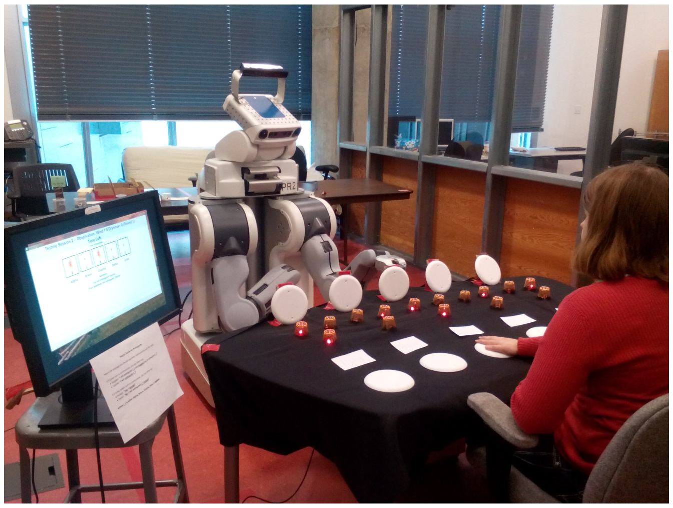

Figure 12: This picture indicates the setup for the embodied robot experiments. Here, a participant works with a PR2 robot to make strategic decisions about how best to extinguish a set of fires.

measures of performance, we measured the reward accumulated during each test session and used the two-sided Mann-Whitney U-test. Differences in the proportion of accepted and rejected suggestions during each test session were evaluated using a two-tailed Fisher's exact test. For subjective measures of performance, we used the two-sided Mann-Whitney U test to compare Likert-scale answers from the post-experiment questionnaire.

8.1 Simulated Robot Experiments

Thirty-six participants were randomly assigned to one of three training conditions: perturbation training using AdaPT, perturbation training using Q-learning or procedural training using Q-learning. We tested for statistically significant differences between perturbation AdaPT and perturbation Q-learning treatments, as well as between perturbation AdaPT and procedural Q-learning treatments. Results are presented in Figure 13.

8.1.1 Perturbation AdaPT vs. Perturbation Q-learning

First, we compared the two algorithms for perturbation training, AdaPT and classical Q-learning, to evaluate the effect of the human's experience of perturbations vs. the effect of the robot's algorithm. We found that perturbation AdaPT-trained teams received statistically significantly higher reward for test task 1, the base task with $w = 0$ and $d = 0$ ($p = 0.0037$), and for test task 3 with $w = 9$ and $d = 2$ ($p = 0.0491$) compared with perturbation Q-learning teams. This was not the case for test task 2, however, most likely because it was similar to the final training session (with a high dryness value) and thus the Q-values reflected the last strategy well. To confirm that perturbation Q-learning only works well for tasks that are similar to the last training session, we ran additional simulation experiments over a range of values and different orders of training sessions. We found that perturbation

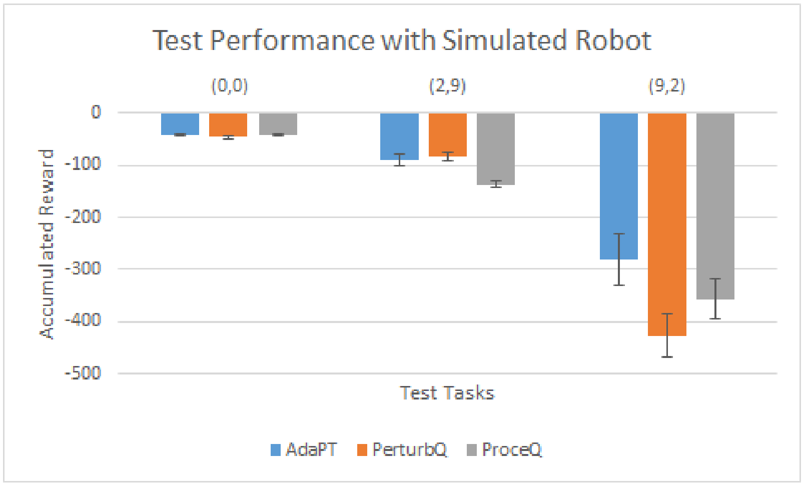

Figure 13: This chart compares the performances of the three different conditions during human subject experiments with simulated robots: perturbation AdaPT, perturbation Q-learning and procedural Q-learning. The y-axis depicts the average accumulated reward received on an execution of the task after the robot simulated on the test task for 1,500 iterations. The x-axis depicts the wind and dryness values of the test tasks (e.g. [2,9] represents a task with $w = 2$ and $d = 9$). Less negative reward denotes better task performance.

Q-learning teams performed well on tasks most similar to the last training session, but never performed statistically significantly better than AdaPT.

We observed a trend toward perturbation AdaPT participants accepting the robot's suggestions more frequently than perturbation Q-learning participants during test task 1 ($p = 0.08$), supporting H1. Subjective measures from the post-experiment questionnaire indicated that participants agreed more strongly that they felt satisfied with the team's performance during test task 1 ($p < 0.01$), lending further support to H1. This result indicates that the algorithm the robot uses, rather than just the human's experience of perturbations, is key to achieving a high level of team performance.

8.1.2 Perturbation AdaPT vs. Procedural Q-learning

We also compared perturbation and procedural training, as has been done previously in human team studies (Gorman et al., 2010), to evaluate the benefit of perturbations during training. For the first test task, which had identical environmental conditions to the procedural training sessions ($w = 0$ and $d = 0$), there were no statistically significant differences in reward between the two training methods.

For the second test task, a substantial variant of the base task and the perturbed training tasks ($w = 2$ and $d = 9$), perturbation AdaPT-trained teams received statistically significantly higher reward than procedural teams (p = 0.0045), supporting H2. In the third test task ($w = 9$ and $d = 2$), perturbation AdaPT teams received higher reward on average, but this difference was not statistically significant.

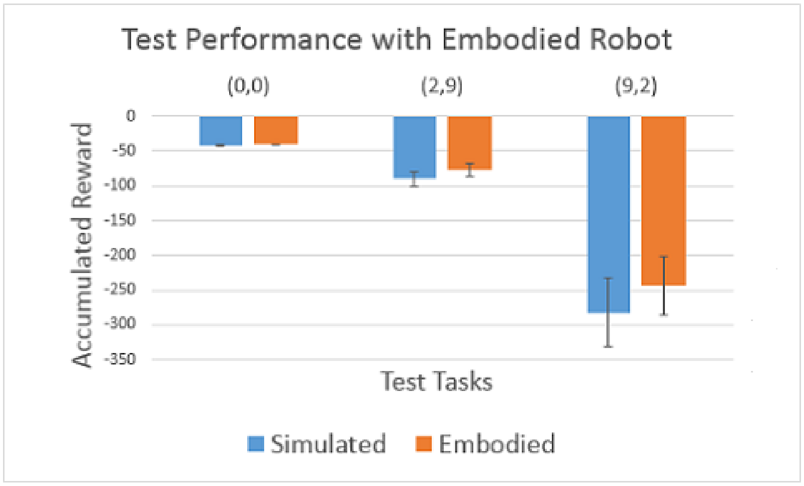

Figure 14: This chart compares the performance of teams that tested with a simulated robot vs. an embodied robot. The average accumulated reward received on an execution of the task with the person after the robot simulated on the test task for 1,500 iterations is depicted along the y-axis. The x-axis displays the wind and dryness values of the test tasks (e.g., [2,9] represents a task with $w = 2$ and $d = 9$). Less negative reward denotes better task performance.

Subjective measures from the post-experiment questionnaire further support H2. Participants in the AdaPT treatment agreed more strongly that the team adapted well to different scenarios compared with participants in the procedural-training treatment ($p = 0.002$). AdaPT participants also agreed more strongly that their teammate offered useful suggestions during testing compared with procedurally trained participants ($p = 0.02$). These results indicate that our perturbation training model can support a human and a robot in generalizing prior experience to new task variants without sacrificing performance on the base task that the procedural teams train on.

8.2 Embodied Robot Experiments

We recruited 12 participants from a university campus and assigned them all to the perturbation AdaPT condition. We compared objective measures of performance between the results from these embodied robot experiments and those from the perturbation AdaPT simulation experiments in order to determine whether people were able to effectively train in simulation and then work with an embodied robot. The results are depicted in Figure 14.

Interestingly, teams that worked with an embodied robot performed better on all three test tasks, on average, than teams that worked with the robot in simulation, thereby supporting H3. While the improvement to performance observed when working with an embodied robot was not statistically significant, it is nonetheless encouraging that no degradation in performance was observed and that simulation training promoted effective human-robot teamwork.

9. Discussion

In this work, we provided the first end-to-end computational framework for perturbation training that enables a robot to co-train with a person during live interactions. Our framework includes the AdaPT algorithm for robot transfer learning, as well as a co-learning method by which the human interacts with the robot in designated intermittent sessions. These sessions allow the human to gain experience by jointly performing tasks with the robot, and enable interaction using a bi-directional communication protocol.

In Section 5, we showed through results from computational experiments that AdaPT learns rapidly during the source tasks, and is thereby able to support real-time interaction with a human. We compared the performance of our algorithm to other state-of-the-art reinforcement and transfer learning techniques: the Policy Reuse in Q-learning (PRQL) algorithm; RBDist, a deep-learning MDP similarity metric; and standard Q-learning; and found that in the two domains tested, the performance of PRQL was highly sensitive to the prior applied. RBDist provided a mechanism for automatically selecting the prior, but the performance was sensitive to specification of additional parameters, such as the number of hidden units.

Results from experiments in the fire extinguishing domain also indicated that RBDist did not uniformly select the best prior. This task involved learning multiple, highly distinct strategies, with significantly lower performance resulting from choosing an inappropriate prior. The performance gap between PRQL with RBDist and AdaPT indicated the potential cost associated with RBDist choosing a suboptimal prior. In contrast, results from the GridWorld task indicated a smaller range in performance for RBDist because strategies were more similar and application of any prior learned through training helped to speed up the learning process. Thus, the expected performance benefit when applying RBDist was domain-dependent, while AdaPT appeared to perform consistently well in both experiment tasks.

AdaPT also yielded benefits in computation time compared with RBDist – an especially desirable feature when applied to co-learning through live interactions with a human. During the fire task, RBDist was slower by a factor of 2.74 with 1 hidden unit, and by a factor of 135.31 with 500 hidden units. For the GridWorld task, RBDist was slower by a factor of 1.87 with 1 hidden unit, and by a factor of 75.76 with 500 hidden units. Rapid learning for new task variants was the primary motivation for developing AdaPT, and enabled us to conduct human subject experiments with the perturbation training model.

We also developed the first end-to-end computational framework for human-robot perturbation training. This framework included an algorithm for source task learning and the transfer of learning to new tasks. We additionally included a co-learning method in which the human interacts with the robot at designated intervals. These interactive sessions allow the human and robot to perform the task together and jointly make decisions using a bi-directional communication protocol.

In our human subject experiments, we first compared two models for perturbation training to isolate the effect of the human's learning from the effect of the robot's learning algorithm. The robot used the AdaPT algorithm under one condition, and standard Q-learning – which we know from computational results consistently performs worse than AdaPT (Section 5) – under the other, with perturbations experienced under both conditions. We found that teams that worked with an agent using AdaPT performed significantly better than a team that worked with an agent using Q-learning ($p = 0.0037$ for test task 1 and $p = 0.0491$ for test task 3). This result indicates that a human is unable to compensate for a robot's poor-performing algorithm; therefore, an appropriate learning algorithm is important for achieving high-level human-robot team performance. AdaPT is able to support this level of performance after training on perturbations with a person.

Next, we compared the performance of teams that trained under perturbations using AdaPT to those that underwent procedural training using Q-learning (i.e., did not experience perturbations during training and instead repeatedly practiced the base task). Our results indicated that the perturbation training framework yielded higher-level team performance on novel tasks than that achieved through a comparable computational model for human-robot procedural training. Specifically, perturbation-trained teams achieved statistically significantly higher reward than procedurally trained teams when tested on the novel task variants most different from the most recently trained task ($p < 0.05$). Interestingly, AdaPT teams performed just as well on the base task during testing as procedurally trained teams; this is in contrast to findings from human team literature, which have indicated that procedural teams typically perform better than perturbation teams on the base task. This is a promising result, as it suggests that AdaPT may provide a framework for adapting to new tasks while also maintaining a high level of performance on previously learned tasks.

Our findings provide the first support for the idea that the relative benefits observed for human team perturbation training can also be realized for human-robot teams. Further, our formalization of the problem statement and computational framework suggest a pathway for future study into alternate learning, transfer and interaction algorithms that improve the ability of human-robot teams to effectively learn complex joint-action tasks through experience of perturbations.

Finally, we demonstrated through results from robot experiments involving a PR2 that human-robot perturbation training in a simulated environment resulted in effective team performance with an autonomous, embodied robot partner that communicated and received commands through speech. This is also an encouraging result, as it indicates that simulation offers an effective and scalable approach to human-robot team training, just as simulation has been widely and successfully used to train human teams (Sandahl et al., 2013; Griswold et al., 2012; Nikolaidis & Shah, 2013).

There are a number of possible extensions to this work. First, our experiments incorporated teams of two, and demonstrated successful human-robot co-learning for problems that were two to three orders of magnitude larger than those used in prior work in human-robot collaborative team training (Nikolaidis & Shah, 2013). However, many real-world domains require the coordination of larger teams. The AdaPT algorithm and perturbation training model can be applied to larger teams; however, like other reinforcement learning methods, the performance of AdaPT will suffer as problem size increases. A number of techniques could be applied to potentially scale to larger teams, including function approximation (Xu, Zuo, & Huang, 2014) and hierarchical reinforcement learning methods (Botvinick, 2012; Barto & Mahadevan, 2003).

A greater number of agents would also require more complex coordination mechanisms. Each agent could no longer be assumed to track the actions of every teammate, and would often only have partial observability of its environment. In order to enable intelligent decision making in the absence of global information, it would potentially be necessary for the robot to perform inference and make predictions about the decisions and actions of other agents, as has been done in prior work (Amir et al., 2014; Van Dyke Parunak, Brueckner, Matthews, Sauter, & Brophy, 2007).

Further, we assumed that the human and robot both communicate at each time step; however, for teams with more agents, it would be infeasible to support communication between all agents prior to every action. Our framework will need to be extended so that robots communicate only pertinent information when necessary, and only to a small subset of agents (Williamson et al., 2009; Roth et al., 2006; Zhang & Lesser, 2013). The MDP model also requires that we explicitly represent the task, agent capabilities and environment. All our experiments included a model for noise in the state evolution; however, there is much uncertainty in the real world, where the structure of the noise may be unknown – and thus cannot be modeled. It is important, therefore, to consider robustness when noise that has not been represented in the model is introduced.

We aim in our follow-on work to conduct human-robot experiments involving more complex, real-world tasks. This initial work was intended to provide proof-of-concept that a human and robot could co-learn joint strategies for application to larger, more complex tasks. Future efforts will demonstrate human-robot perturbation training for tasks such as complex manipulation or collaborative search and rescue.

10. Conclusion

We designed and evaluated a computational learning model for perturbation training of human-robot teams, motivated by human team-training studies. We first presented a problem definition for perturbation training, which involved a transfer learning component, a co-learning framework and a communication protocol. Next, we developed a learning algorithm, AdaPT, that augments the PRQL algorithm (Fernández et al., 2010) to learn faster for new task variants. We found that AdaPT is able to more quickly and robustly learn during new tasks compared to prior work. We further included a human-robot co-learning framework and a bidirectional communication protocol to form a computational model for perturbation training. In human subject experiments with a simulated robot, we observed that perturbation-trained teams using AdaPT outperformed perturbation-trained teams using Q-learning with regard to both objective ($p = 0.0037$ for test task 1 and $p = 0.0491$ for test task 3) and subjective measures of performance for multiple task variants. This indicates that the robot's algorithm, rather than perturbations alone, is key to achieving a high level of team performance. Perturbation AdaPT teams also did not sacrifice performance of the base task when compared with procedurally trained teams. Finally, we demonstrated in robot experiments with a PR2 that human-robot training in a simulation environment using AdaPT resulted in effective team performance with an autonomous, embodied robot partner.

Acknowledgments

We thank Dr. Eric Eaton and Dr. Haitham Ammar for their help in implementing the RBDist measure. We thank Karthik Narasimhan, Vaibhav Unhelkar, Jorge Perez, and the Interactive Robotics Group for their insightful discussions and comments. We also thank NDSEG for funding this research.

References

Agogino, A., & Tumer, K. (2005). Reinforcement learning in large multi-agent systems. In *AAMAS-05 Workshop on Coordination of Large Scale Multi-Agent Systems. Utrecht, Netherlands*. Citeseer.

Akgun, B., Cakmak, M., Yoo, J. W., & Thomaz, A. L. (2012). Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pp. 391–398. ACM.

Alexandrova, S., Cakmak, M., Hsiao, K., & Takayama, L. (2012). Robot programming by demonstration with interactive action visualizations. In *Robotics Science and Systems*.

Amir, O., Grosz, B. J., & Stern, R. (2014). To share or not to share? the single agent in a team decision problem. *Models and Paradigms for Planning under Uncertainty: a Broad Perspective*, 19.

Ammar, H. B., Eaton, E., Ruvolo, P., & Taylor, M. E. (2015). Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment. In *Proc. of AAAI*.

Ammar, H. B., Eaton, E., Taylor, M. E., Mocanu, D. C., Driessens, K., Weiss, G., & Tuyls, K. (2014). An automated measure of MDP similarity for transfer in reinforcement learning. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Barto, A. G., & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, *13*(1-2), 41–77.

Botvinick, M. M. (2012). Hierarchical reinforcement learning and decision making. *Current opinion in neurobiology*, *22*(6), 956–962.

Bowling, M., & Veloso, M. (2000). An analysis of stochastic game theory for multiagent reinforcement learning. Tech. rep., DTIC Document.

Brunskill, E., & Li, L. (2013). Sample complexity of multi-task reinforcement learning. *arXiv preprint arXiv:1309.6821*.

Busoniu, L., Babuska, R., & De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, *38*(2), 156–172.

Carroll, J. L., & Seppi, K. (2005). Task similarity measures for transfer in reinforcement learning task libraries. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, Vol. 2, pp. 803–808. IEEE.

Chao, C., & Thomaz, A. L. (2012). Timing in multimodal turn-taking interactions: Control and analysis using timed Petri-Nets. *Journal of Human-Robot Interaction*, *1*(1).

Chernova, S., & Veloso, M. (2010). Confidence-based multi-robot learning from demonstration. *International Journal of Social Robotics*, *2*(2), 195–215.

Cohen, M. S., Freeman, J. T., & Wolf, S. (1996). Metarecognition in time-stressed decision making: Recognizing, critiquing, and correcting. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *38*(2), 206–219.

Eaton, E., & DesJardins, M. (2006). Knowledge transfer with a multiresolution ensemble of classifiers. In *ICML Workshop on Structural Knowledge Transfer for Machine Learning*.

Eaton, E., desJardins, M., & Lane, T. (2008). Modeling transfer relationships between learning tasks for improved inductive transfer. In *Machine Learning and Knowledge Discovery in Databases*, pp. 317–332. Springer.

Fachantidis, A., Partalas, I., Taylor, M. E., & Vlahavas, I. (2015). Transfer learning with probabilistic mapping selection. *Adaptive Behavior*, *23*(1), 3–19.

Fan, X., & Yen, J. (2005). Conversation pattern-based anticipation of teammates' information needs via overhearing. In *Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*, pp. 316–322. IEEE.

Fernández, F., García, J., & Veloso, M. (2010). Probabilistic policy reuse for inter-task transfer learning. *Robotics and Autonomous Systems*, *58*(7), 866–871.

Goldman, C. V., & Zilberstein, S. (2003). Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pp. 137–144. ACM.

Gorman, J. C., Cooke, N. J., & Amazeen, P. G. (2010). Training adaptive teams. *Human Factors: The Journal of the Human Factors and Ergonomics Society*.

Griffith, S., Subramanian, K., Scholz, J., Isbell, C., & Thomaz, A. L. (2013). Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2625–2633.

Griswold, S., Ponnuru, S., Nishisaki, A., Szyld, D., Davenport, M., Deutsch, E. S., & Nadkarni, V. (2012). The emerging role of simulation education to achieve patient safety: translating deliberate practice and debriefing to save lives. *Pediatric Clinics of North America*, *59*(6), 1329–1340.

Guestrin, C., Lagoudakis, M., & Parr, R. (2002). Coordinated reinforcement learning. In *ICML*, Vol. 2, pp. 227–234.

Hawkins, K. P., Bansal, S., Vo, N. N., & Bobick, A. F. (2014). Anticipating human actions for collaboration in the presence of task and sensor uncertainty. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 2215–2222. IEEE.

Kim, S. K., & Byun, S. N. (2011). Effects of crew resource management training on the team performance of operators in an advanced nuclear power plant. *Journal of nuclear science and technology*, *48*(9), 1256–1264.

Klein, G. A. (1989). Do decision biases explain too much?. *Human Factors Society Bulletin*, *32*(5), 1–3.

Knox, W. B., & Stone, P. (2009). Interactively shaping agents via human reinforcement: The TAMER framework. In *Proceedings of the fifth international conference on Knowledge capture*, pp. 9–16. ACM.

Konidaris, G., & Barto, A. G. (2007). Building portable options: Skill transfer in reinforcement learning. In *IJCAI*, Vol. 7, pp. 895–900.

Kono, H., Murata, Y., Kamimura, A., Tomita, K., & Suzuki, T. (2016). Knowledge co-creation framework: Novel transfer learning method in heterogeneous multi-agent systems. In *Distributed Autonomous Robotic Systems*, pp. 389–403. Springer.

Leonard, M., Graham, S., & Bonacum, D. (2004). The human factor: the critical importance of effective teamwork and communication in providing safe care. *Quality and Safety in Health Care*, *13*(suppl 1), i85–i90.

Mahmud, M., & Ramamoorthy, S. (2013). Learning in non-stationary MDPs as transfer learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 1259–1260. International Foundation for Autonomous Agents and Multiagent Systems.

Mann, T. A., & Choe, Y. (2012). Directed exploration in reinforcement learning with transferred knowledge.. In *EWRL*, pp. 59–76.

Marks, M. A., Sabella, M. J., Burke, C. S., & Zaccaro, S. J. (2002). The impact of cross-training on team effectiveness. *Journal of Applied Psychology*, *87*(1), 3.

Marshall, S. D., & Flanagan, B. (2010). Simulation-based education for building clinical teams. *Journal of Emergencies, Trauma and Shock*, *3*(4), 360.

Mohseni-Kabir, A., Rich, C., Chernova, S., Sidner, C. L., & Miller, D. (2015). Interactive hierarchical task learning from a single demonstration. In *Human-Robot Interaction (HRI) 2015*. IEEE.

Newell, A., Simon, H. A., et al. (1972). *Human problem solving*, Vol. 104. Prentice-Hall Englewood Cliffs, NJ.

Niekum, S., Osentoski, S., Konidaris, G., Chitta, S., Marthi, B., & Barto, A. G. (2014). Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 0278364914554471.

Nikolaidis, S., & Shah, J. (2013). Human-robot cross-training: computational formulation, modeling and evaluation of a human team training strategy. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pp. 33–40. IEEE Press.

Oudah, M., Babushkin, V., Chenlinangjia, T., & Crandall, J. W. (2015). Learning to interact with a human partner. In *Human-Robot Interaction (HRI) 2015*. IEEE.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, *22*(10), 1345–1359.

Parisotto, E., Ba, J. L., & Salakhutdinov, R. (2016). Actor-mimic: Deep multitask and transfer reinforcement learning..

Puterman, M. L. (2009). *Markov decision processes: discrete stochastic dynamic programming*, Vol. 414. John Wiley & Sons.

Roth, M., Simmons, R., & Veloso, M. (2005). Decentralized communication strategies for coordinated multi-agent policies. In *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III*, pp. 93–105. Springer.

Roth, M., Simmons, R., & Veloso, M. (2006). What to communicate? Execution-time decision in multi-agent POMDPs. In *Distributed Autonomous Robotic Systems 7*, pp. 177–186. Springer.

Ruvolo, P., & Eaton, E. (2013). Ella: An efficient lifelong learning algorithm.. *ICML (1)*, *28*, 507–515.

Rybski, P. E., Yoon, K., Stolarz, J., & Veloso, M. M. (2007). Interactive robot task training through dialog and demonstration. In *Human-Robot Interaction (HRI), 2007 2nd ACM/IEEE International Conference on*, pp. 49–56. IEEE.

Sandahl, C., Gustafsson, H., Wallin, C.-J., Meurling, L., Øvretveit, J., Brommels, M., & Hansson, J. (2013). Simulation team training for improved teamwork in an intensive care unit. *International journal of health care quality assurance*, *26*(2), 174–188.

Shen, J., Lesser, V., & Carver, N. (2003). Minimizing communication cost in a distributed Bayesian network using a decentralized MDP. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pp. 678–685. ACM.

Sutton, R. S., & Barto, A. G. (1998). *Introduction to reinforcement learning*. MIT Press.

Talvitie, E., & Singh, S. P. (2007). An experts algorithm for transfer learning. In *IJCAI*, pp. 1065–1070.

Tanaka, M., & Okutomi, M. (2014). A novel inference of a restricted Boltzmann machine. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pp. 1526–1531. IEEE.

Taylor, A., Duparic, I., Galván-López, E., Clarke, S., & Cahill, V. (2013). Transfer learning in multi-agent systems through parallel transfer..

Taylor, M. E., Jong, N. K., & Stone, P. (2008). Transferring instances for model-based reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases*, pp. 488–505. Springer.

Taylor, M. E., Kuhlmann, G., & Stone, P. (2007). Accelerating search with transferred heuristics. In *ICAPS Workshop on AI Planning and Learning*.

Taylor, M. E., Kuhlmann, G., & Stone, P. (2008). Autonomous transfer for reinforcement learning. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1*, pp. 283–290. International Foundation for Autonomous Agents and Multiagent Systems.

Taylor, M. E., & Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *The Journal of Machine Learning Research*, *10*, 1633–1685.

Taylor, M. E., Stone, P., & Liu, Y. (2005). Value functions for rl-based behavior transfer: A comparative study. In *Proceedings of the National Conference on Artificial Intelligence*, Vol. 20, p. 880. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Taylor, M. E., Suay, H. B., & Chernova, S. (2011). Using human demonstrations to improve reinforcement learning. In *AAAI Spring Symposium: Help Me Help You: Bridging the Gaps in Human-Agent Collaboration*.

Torrey, L., & Shavlik, J. (2009). Transfer learning. *Handbook of Research on Machine Learning Applications. IGI Global*, *3*, 17–35.

Van Dyke Parunak, H., Brueckner, S., Matthews, R., Sauter, J., & Brophy, S. (2007). Real-time agent characterization and prediction. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, p. 278. ACM.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, *8*(3-4), 279–292.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. Ph.D. thesis, University of Cambridge.

Williamson, S. A., Gerding, E. H., & Jennings, N. R. (2009). Reward shaping for valuing communications during multi-agent coordination. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 641–648. International Foundation for Autonomous Agents and Multiagent Systems.

Wilson, A., Fern, A., Ray, S., & Tadepalli, P. (2007). Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th international conference on Machine learning*, pp. 1015–1022. ACM.

Wu, F., Zilberstein, S., & Chen, X. (2009). Multi-agent online planning with communication. In *ICAPS*.

Xu, X., Zuo, L., & Huang, Z. (2014). Reinforcement learning algorithms with function approximation: Recent advances and applications. *Information Sciences*, *261*, 1–31.

Xuan, P., Lesser, V., & Zilberstein, S. (2000). Communication in multi-agent Markov decision processes. In *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on*, pp. 467–468. IEEE.

Xuan, P., Lesser, V., & Zilberstein, S. (2001). Communication decisions in multi-agent cooperation: Model and experiments. In *Proceedings of the fifth international conference on Autonomous agents*, pp. 616–623. ACM.

Yen, J., Yin, J., Ioerger, T. R., Miller, M. S., Xu, D., & Volz, R. A. (2001). Cast: Collaborative agents for simulating teamwork. In *International Joint Conference on Artificial Intelligence*, Vol. 17, pp. 1135–1144. LAWRENCE ERLBAUM ASSOCIATES LTD.

Zhang, C., & Lesser, V. (2012). Coordinated multi-agent learning for decentralized POMDPs. In *Proceedings of the Seventh Annual Workshop on Multiagent Sequential Decision-Making Under Uncertainty (MSDM)*.

Zhang, C., & Lesser, V. (2013). Coordinating multi-agent reinforcement learning with limited communication. In *Proceedings of the 2013 international conference on Au-*

tonomous agents and multi-agent systems, pp. 1101–1108. International Foundation for Autonomous Agents and Multiagent Systems.

Zhang, C., & Lesser, V. R. (2011). Coordinated multi-agent reinforcement learning in networked distributed POMDPs. In *AAAI*.