Abstract

Supportive Behaviors for Human-Robot Teaming

Bradley Hayes

2016

While robotics has made considerable strides toward more robust and adaptive manipulation, perception, and planning, robots in the near future are unlikely to be as dexterous, competent, and versatile as human workers. Rather than try to create fully autonomous systems that accomplish tasks independently, a more practical approach is to construct robots that work alongside people. This allows human and robot workers to concentrate on the tasks for which they are each best suited, while simultaneously providing the capability to assist each other during tasks that one worker lacks the ability to complete independently in a safe or maximally proficient manner. Human-robot teaming advances have the potential to extend applications of autonomous robots well beyond their current, limited roles in factory automation settings. Much of modern robotics remains inapplicable in many domains where tasks are either too complex, beyond modern hardware limitations, too sensitive for non-human completion, or too flexible for static automation practices. In these situations human-robot teaming can be leveraged to improve the efficiency, quality-of-life, and safety of human partners.

In this thesis, I describe algorithms that can create collaborative robots that can provide assistance when useful, remove dull or undesirable responsibilities when possible, and assist with dangerous tasks when feasible. In doing so, I present a novel method for autonomously constructing hierarchical task networks that factor complex tasks in ways that make them approachable by modern planning and coordination algorithms. In particular, within these complex cooperative tasks I focus on facilitating collaboration between a lead worker and robotic assistant within a shared space, defining and investigating a class of actions I term supportive behaviors: actions that serve to reduce the cognitive or kinematic complexity of tasks for teammates. The majority of contributions within this work center around discovering, learning, and executing these types of behaviors in multi-agent domains

with asymmetric authority. I provide an examination of supportive behavior learning and execution from the perspective of task and motion planning, as well as that of learning directly from interactions with humans. These algorithms provide a collaborative robot with the capability to anticipate the needs of a human teammate and proactively offer help as needed or desired. This work enables to creation of robots that provide tools just-in-time, robots that alter workspaces to make more optimal task orderings more obvious and more feasible, and robots that recognize when a user is delayed in a complex task and offer assistance.

Combining these algorithms provides a basis for a robot with both a capacity for rich task comprehension and a theory of mind about its collaborators, enabling methods to allow such a robot to leverage knowledge it acquires to transition between the role of learner, able assistant, and informative instructor during interactions with teammates.

Supportive Behaviors for Human-Robot Teaming

A Dissertation
Presented to the Faculty of the Graduate School
of
Yale University
in Candidacy for the Degree of
Doctor of Philosophy

by
Bradley Hayes

Dissertation Director: Brian Scassellati

May, 2016

Contents

Acknowledgements **vi**

1 Introduction **1**

1.1 Background and Related Work 5

1.1.1 Learning from Demonstration 6

1.1.2 Active Learning 8

1.1.3 Task Representation and Shared Mental Models 8

1.1.4 Interactive Policy Learning 10

1.2 Major Challenges in Human-Robot Collaboration 13

1.2.1 Important Problems in Shared-Environment Collaboration 13

1.2.2 Achieving Bi-Directional Intention Recognition between Humans and Robots 15

1.2.3 Role Selection When Working with Human Teammates 18

1.2.4 Evaluating Trade-offs Between Task Execution Optimality and Co-worker Preferences 20

1.2.5 Evaluating Performance During Live Collaboration 22

1.3 Robot Platforms 24

1.4 Contributions 25

2 Hierarchical Task Networks for Human-Robot Collaboration **27**

2.1 A Hybrid Approach to HTN Creation from Demonstrations and Planning . 32

2.1.1 Task Discovery Through Constraint Analysis 33

2.1.2 Task Encoding and Action Representation 34

2.1.3 Conjugate Task Graph . . . 35
2.1.4 CC-HTN Generation . . . 39
2.1.5 Hierarchical Ambiguity . . . 40
2.1.6 Algorithm Runtime Analysis . . . 42
2.1.7 Applications and Evaluation . . . 44
2.1.8 Summary . . . 48
2.2 Discovering Task Constraints through Active Learning . . . 49
2.2.1 Motivation . . . 50
2.2.2 Approach . . . 52
2.2.3 Learning Environment . . . 53
2.2.4 Activity Description . . . 55
2.2.5 Learning System . . . 56
2.2.6 Query Strategies . . . 58
2.2.7 Experiment Design . . . 59
2.2.8 Results . . . 63
2.2.9 Individual Training . . . 63
2.2.10 Joint Training . . . 64
2.2.11 Discussion and Conclusion . . . 67

3 Supportive Behaviors **69**
3.1 A Taxonomy of Supportive Behaviors . . . 70
3.2 A Task and Motion Planning Approach . . . 72
3.2.1 Introduction . . . 72
3.2.2 Related Work . . . 74
3.2.3 Task and Motion Planning Domain . . . 75
3.2.4 Supportive Behavior Generation . . . 76
3.2.5 Formalizing the Supportive Behavior Problem . . . 77
3.2.6 Plan weight determination . . . 79
3.2.7 Environment state analysis . . . 81
3.2.8 Assumptions and Weaknesses . . . 82

3.2.9 Evaluation . . . 83
3.2.10 Summary . . . 89
3.3 Learning from Demonstration and Natural Language . . . 90
3.3.1 Approach . . . 91
3.3.2 Policy transfer for supportive behaviors . . . 93
3.3.3 Associating supportive behaviors with subtasks . . . 94
3.3.4 Defining initiation and termination conditions . . . 95
3.3.5 A metric for team fluency . . . 102
3.4 Social Cues for Task Understanding and Role Evolution . . . 104
3.4.1 Related Work . . . 108
3.4.2 Approach . . . 109
3.4.3 Task Structure and Execution . . . 112
3.4.4 Computing Social Force . . . 113
3.4.5 Role Transitions Leveraging Social Force . . . 114
3.4.6 Summary . . . 117
3.5 Enhancing Agent Safety and Learning through Supportive Behaviors . . . 120
3.5.1 Motivation . . . 120
3.5.2 Preliminaries . . . 123
3.5.3 A Model of Safety . . . 124
3.5.4 A Model of Danger . . . 126
3.5.5 Experiments . . . 127
3.5.6 Evaluation Criteria . . . 129
3.5.7 Results . . . 130
3.5.8 Discussion and Summary . . . 133

4 Conclusion and Future Directions **136**

4.1 Summary of Contributions . . . 136
4.2 Future Work . . . 137
4.2.1 Resolving uncertainty in collaborative tasks . . . 137
4.2.2 Task Monitoring and State Estimation . . . 138

4.2.3 Converging upon shared mental models 140

List of Figures

1.1 An IKEA Lätt chair, used as a representative assembly task throughout this thesis. 15

1.2 Robot platforms used within the collaborative task execution research in this thesis. 24

2.1 Collaboration relies on complex planning and intention recognition capabilities. For many tasks, especially within the assembly and cooking domains the HTN evaluations are based upon, hierarchical models are required to identify and solve tractable sub-problems. 31

2.2 Task graph (top) and its conjugate (bottom). In the task graph, environment states are encoded in vertices and edges are labeled with their corresponding actions. In the conjugate, subgoals are represented as vertices and edges are labeled with environmental prerequisites. 36

2.3 Incremental execution of Algorithm **??**. For clarity, I omit edge prerequisite conditions. The internal node labels on the final HTN were manually added for clarity, illustrating the implied logic behind the action grouping. 37

2.4 Mean computation times for estimating an agent's last completed subgoal using HMMs, as a function of task complexity. Timings were measured using a single thread of an Intel i7-3930K CPU. The responsiveness required in collaborative task execution mandates high frequency state estimation capabilities. In most cases, the CC-HTN model completed its computation faster than 1Hz. 41

2.5 Average number of edges in generated sequential manipulation tasks as a function of subgoal count. The CC-HTN encapsulates much of the transition complexity in the task, causing a dramatic reduction in the amount of edges required to represent task substructure. This simplification of task dynamics allows for more rapid computations in role selection and intention recognition. 42

2.6 Task MDP discovery results on a dataset of 25 food preparation tasks with a primitive action set size of 39. Q-function transfer aggregates Q-functions from known task MDPs to bias action selection during exploration. CC-HTN Transfer uses Q-function transfer with macro actions learned from the CC-HTNs of other tasks in the data set. 47

2.7 Collaborative Workbench platform with experimental setup, used to learn task structure from demonstrations of construction activities. 51

2.8 Visualization of graph transforms on an everyday task. For this task, the knife must be acquired prior to chopping, though the order that the carrots or celery is chopped does not matter. 55

2.9 Construction activity task hierarchies 60

2.10 Active learning strategies compared across tasks and data sets. 62

2.11 Results for simulated active learning strategies, averaged over a set of 12 kinesthetically trained food preparation tasks. 64

3.1 A graphical representation of the supportive behavior generation pipeline. . 76

3.2 Plot of the second weighting scheme enumerated below. This graph shows weights assigned to plan improvements (and impediments) as a function of their optimality versus the best known plan. 80

3.3 One sample scenario used in the evaluation. In this task, the lead agent must create a circuit using two power sources wired in series to drive an LED on a switched circuit. Random configurations of objects were utilized to create a variety of starting conditions. 82

3.4 Example of a SnapCircuits solution. Snap buttons indicate valid connection points between components, which must be joined to form circuits in a manner that satisfies the implicit 3D spatial constraints imposed by the pieces available and the desired goal circuit function. 84

3.5 Average makespan over multiple random initial environmental configurations of the base task. Adding a support robot provided substantial task completion time improvement even though the supportive agent could not directly contribute towards the task completion. 86

3.6 Mean planning duration of the lead agent for solving various SnapCircuits TAMP problems. In multi-agent scenarios, due to the decoupled nature of my approach, the agents had to replan when resource or spatial conflicts emerged (as compared to no replanning occurring during the unsupported conditions). 87

3.7 The Collaborative Workbench used in the proof-of-concept implementation. The left monitor mirrors the tablet's display, showing live information about the task tree throughout task execution and allowing human participants to claim roles within the system. The right monitor displays a simulated environment mirroring the real world, along with social force fields as they are generated from user motion. 107

3.8 Realtime visualization of a social force particle field generated from the user's motion patterns, rendered over point cloud data from the workbench sensors. Social force magnitude is represented by the blue channel, with the intensity of force increasing as particles range from black to blue. The estimated position of the user's hand is represented by a red square, shown under the gloved hand. 111

3.9 Full demonstration of the task by the robotic agent. When multiple agents participate, it is possible to execute the subtask that joins the blue and green bases and the subtask that moves the red block into position in parallel. . . 112

3.10 An example of the "student" behavior, achieved by attractive social force. In the first panel, the robot is waiting for guidance before choosing an action. In the second panel, the user has gestured near the blue block, exerting social force in the goal region of "Locate Blue Block". The red box around the skill name indicates that robot intends to complete this action. The final panel shows the action as 'completed' in the task tree visualization (green box). . 118

3.11 An example of the "peer" behavior, achieved by repulsive social force. In the first panel, the robot intends to complete "Locate Blue Block". The second panel shows an interruption of the action by a human user. Upon detecting strong social force in its current skill's goal region, the robot aborts its execution and chooses a different, non-conflicting skill. The third panel shows the robot continuing to exercise conflict-avoidant behavior with the user. 119

3.12 An example of the "instructor" behavior, achieved by a using social force as a trigger. The first panel shows the robot evaluating available actions to teach. The second panel shows the robot demonstrating part of the "Pickup Red Block" subtask without actually completing it. In the third panel, the user has picked up the red block as previously pointed to by the robot, triggering the robot to mark the task as complete. 119

3.13 A map used in the experiments. The environment consists of open spaces, walls, staircases, tables, candles, and toy bins. 128

3.14 The average normalized damage (negative reward) accumulated by novice agents of various levels of goal-directedness, both with and without the assistance of a caregiver robot. Error bars represent one standard deviation from the mean. A value of 1.0 on this graph corresponds to the maximum damage received in a single episode across all agent types and conditions. Novices followed an ϵ-greedy policy to their goal, determining the frequency of choosing optimal or random exploratory actions. 131

3.15 The average number of timesteps (maximum 200) before the novice agent incurred damage above the stoppage threshold. Error bars represent one standard deviation from the mean. Values of 200 indicate that the agent completed the entire interaction without incurring damage above the termination threshold. 133

3.16 The average number of unique grid cells the novice visited per episode. Error bars represent one standard deviation from the mean. The number of cells visited is used as a metric to evaluate the amount of environmental knowledge gained by the novice agent over the course of an episode (up to 200 timesteps). 134

List of Tables

3.1 Mean normalized damage (negative reward) accrued by various novice agents with and without caregiver intervention 132

3.2 Average number of timesteps until threshold damage with and without caregiver intervention 132

3.3 Average number of cells explored with and without caregiver intervention . 135

Acknowledgements

I owe a great deal to all who have supported me throughout my academic career to date, culminating in the completion of this thesis. In particular, I am grateful for the tremendous support provided by my advisor, Brian Scassellati. Throughout my time at Yale, he has enthusiastically encouraged me to take chances with novel research, taught me how to effectively communicate my work, and provided me with many opportunities to further my development as a researcher. He has been an incredible resource, always making time to meet, sending me to conferences, and providing feedback on drafts of my papers at all hours of the evening. His mentorship and efforts in taking an active role within my professional development were invaluable.

My time at Yale would have been much less enriching if not for my postdoc and graduate student colleagues. I am indebted to them for the productive discussions and perspectives they have provided me. In particular, I thank Cindy Bethel, Dave Feil-Seifer, Iolanda Leite, André Pereira, Kate Tsui, Corina Grigore, Alex Litoiu, Aditi Ramachandran, and Sarah Strohkorb for being great sources of enrichment and friendship within the Social Robotics Lab.

I would also like to thank my fantastic friends and family for their support, both during my time at Yale and for all the events leading up to my enrollment. I thank my brother Brandon for his constant support and encouragement, for always making time to talk, and for his willingness to get me away from my work on a moment's notice when things got overwhelming. To Steve, Ruth, Jim, and Kathy, I could not ask for a more supportive and encouraging family, and I am continually grateful to you all. I especially thank Alex Cerjan and Katrina Sliwa for their great friendship and companionship through countless

adventures at Yale. I certainly couldn't ask for better friends. Relatedly, I would also like to thank Dan Winograd-Cort, Shu-chun Weng, David Costanzo, Valerie Morley, Fiona Hecksher, Jane Widness, the Yale Graduate Ice Hockey team, and everyone at Soulcraft Brazilian Jiu-Jitsu for their incredible support and friendship.

I also thank the Office of Naval Research and the National Science Foundation for funding this work.

Chapter 1

Introduction

While robotics has made considerable strides toward more robust and adaptive manipulation, perception, and planning, robots in the near future are unlikely to be as dexterous, competent, and versatile as human workers. Rather than try to create fully autonomous systems that accomplish tasks independently, a more practical approach is to construct robots that work alongside people. This allows human and robot workers to concentrate on the tasks for which they are each best suited, while simultaneously providing the capability to assist each other during tasks that one worker lacks the ability to complete independently in a safe or maximally proficient manner. Such a collaborative robot might hand you a screwdriver just when it is needed, stabilize a piece of lumber while a human drills at one end, clean up a shared workspace in anticipation of the next task to be completed, alert workers to safety hazards due to the fact that an assembly is wobbly, or help maintain team knowledge of task state.

Developing these robot co-workers falls within the domain of sequential decision making under uncertainty, a fundamental problem within the field of artificial intelligence. Even in the case where a single worker is solving a sequential manipulation problem with uncertainty, environmental dynamics and complex manipulations can quickly push even seemingly trivial problems into the realm of intractability. While existing techniques for dealing with uncertainty have increased the scope of accessible problems, it remains true that uncertainty exponentially increases the difficulty of these already challenging domains.

Within collaborative settings, the introduction of additional agents dramatically in-

creases the stochasticity of the system, increasing the difficulty of finding suitable solutions in kind. Teamwork in general is situated within a more difficult class of sequential decision making problems than single agent scenarios, as agent actions may be coupled in non-obvious or distant ways. Further complicating matters, multi-agent teams are evaluated based on metrics in both long-term (e.g., task completion time, solution quality) and short-term (e.g., agent safety, teammate idle time) time horizons. Effective human teams participate in a constantly synchronizing process of coordinated behaviors, with policy, action, and role selection being dependent on social cues, task understanding, physical capabilities, and each collaborator's understanding of the others' proficiencies. To achieve fluent human-robot teaming, algorithms and systems must be developed that leverage these bases of effective collaboration.

Further contributing to the complexity of teaming, collaborators typically have differing proficiencies and capabilities. Particularly within the context of today's robotics, it is extremely rare for a robot to have the tooling or ability to perform all subcomponents of a complex task. Robots that possess such capabilities are generally designed with a particular task specialization in mind and do not generalize well to new tasks. Robots in the home will not be able to utilize this design strategy, given the multitude of task contexts and environments they may be expected to operate in. Collaborative robots in factory settings may also be unable to benefit from hyper-specialization, as the mass production required to make them economically viable for small businesses effectively eliminates the ability to focus on a single class of use cases.

These potentially non-overlapping sets of capabilities introduce additional complications to the multi-agent cooperation problem: not every agent will be able to complete pieces of the task at any given stage. This does not necessarily mean that agents who are unable to progress the task directly do not have immediate value, as supportive roles may be available for such teammates to assume. Accordingly, actions taken in collaborative activities can generally be segmented into two classes: primary and supportive. Primary actions are considered to be those that directly contribute towards achieving the goals of the task. While a primary action may involve using a hammer to drive a nail into a board, a supportive action may entail holding the board steady during the interaction. Supportive actions, while

not contributing directly towards goal attainment, facilitate task completion by reducing the cognitive load or kinematic difficulty for other agents' primary actions.

This thesis addresses a valuable instance of the sequential decision making problem within mixed human-robot teams, the problem of learning and executing supportive behaviors. Integrating human agents into the multi-agent problem adds a great deal of uncertainty, as the plans, policies, and a multitude of other internal state features of people are not observable. Similarly, it is non-trivial to ensure that these same aspects of a robot are transparent and understandable to human collaborators. Throughout this thesis, I characterize the problem domain using the Markov Decision Process (MDP) framework, but defer formalizing this representation until the contributions presented within later chapters. Intuitively, this framework models a system (such as a robot building a circuit), as being in a discretely specifiable state at each timestep. The system moves between world states through the execution of actions (such as picking up a resistor). It is the job of the agents within such a system to reliably choose sequences of actions ('execute a policy') such that the system both remains in "good" regions of state space (e.g., those where the circuit is not shorted and co-workers are uninjured) and, within sequential manipulation tasks, is progressing the system towards the ultimate task goal (such as a completed, functional circuit). This framework is a popular choice for robotics domains, as it is capable of representing an extremely broad and diverse array of application domains [1–8].

Robots have the potential to revolutionize many domains, spanning all manner of industry from manufacturing to healthcare. Today's robots operate autonomously with speed and precision within tightly controlled, isolated environments. The next step in personal and industrial robotics is to move robots out of isolation and into collaborative relationships, operating side-by-side with human personnel. As demonstrated in recent years, the introduction of these systems to real-world environments requires a substantial engineering effort with a carefully planned interaction methodology, so that human operators can effectively utilize robotic resources. I include in this section an exposition of what I believe to be some of the most important research questions yet to be deeply and fully addressed within shared-environment collaborative robotics. I highlight relevant challenges associated with these research questions, as well as identify existing and future work in the field.

Collaborative operation is ripe for exploration and innovation, opening the possibility for widespread adoption of robots into problem domains for which they may currently be perceived as unsuitable or unready. Even with the current state-of-the-art in skill acquisition and execution, robots have difficulty performing at their full potential when removed from the typically well-controlled environment of the lab.

In this thesis, I use the term *collaborative task execution* to describe an agent autonomously performing a task either collaboratively with or in the presence of other agents, while respecting any associated social roles and divisions of responsibility. Within this definition, I focus exclusively on fully autonomous robots and humans working as a team, as opposed to teleoperated robots. A *skill* is defined as a temporally extended action similarly to options in reinforcement learning [9], and is assumed to be specifiable by a set of preconditions, expected post-conditions, and goal states. I define a *task* as a planning problem with full specification of participating agents and goals, and a *plan* as an arrangement of skills that will result in a successful execution of that task. A plan constructed to include multiple agents may include parallel branches of skill execution, subject to the ordering constraints of the individual skills involved.

Requirements for robot control systems that are capable of engaging in collaborative behaviors with humans are incredibly complex. Such systems require many components to operate safely and properly, often leveraging the state-of-the-art in LfD, intention detection, speech recognition, non-verbal communication, planning systems, physical manipulation, and more. Attempting to build a socially collaborative system forces the designer to address issues of safety, user modeling, environment sensing, and various teamwork-related human factors in addition to the already numerous technical challenges inherent to robotics applications involving real-world manipulation [10].

The primary contributions of this thesis can be found in chapters 2 and 3, where I introduce novel methods of autonomously achieving task comprehension through legible, hierarchical abstractions, as well as mechanisms by which a robot teammate can learn and execute supportive behaviors to overcome these challenges. In the sections that follow I provide a review of recent work, giving a brief introduction to Learning from Demonstration, Active Learning, and Task Acquisition before identifying relevant, important challenges

within human-robot collaboration.

1.1 Background and Related Work

Most existing work relevant to human-robot collaboration has focused on three topics: skill acquisition (motor primitive learning) [6], role management (scheduling) [11–13], and interpreting social signals (communication) [7]. Developing capable systems that are accessible to non-roboticists is a primary concern for collaborative robotics, so many existing methods within HRI research are targeted towards engaging non-technical users. Further, shared-space human-robot collaboration can be a powerful enabler for skill transfer between humans and robots, making this an especially attractive domain for study.

In particular, collaborative tasks provide great opportunities for applications of learning from demonstration. As programming robots explicitly to perform complex skills can be incredibly time-consuming and difficult, researchers have sought more intuitive, accessible methods of skill acquisition. Drawing inspiration from humans' ability to readily acquire new abilities from observations of others, an assortment of algorithms that allow robots to convert observations or teleoperated sessions into skill training data have been developed. Learning from Demonstration includes mechanisms for enabling skill transfer between humans and robots, producing systems that learn from skill executions led by humans or other agents [14].

As an example, consider the task of moving a glass of water from one table to another. For a robot to accomplish this successfully, it is not enough to merely pick up the glass from the source table and place it on the destination table. Implicit to this problem is the constraint that the robot must always orient the glass upright, otherwise the glass contents will pour out. There may also be other difficult constraints imposed, such as being able to recognize acceptable regions on the destination table. A programmer attempting to learn this through a standard reinforcement learning approach might be able to encode the task success criteria, but the robot may never be able to execute enough trials to learn a satisfactory action policy. With learning from demonstration, an expert can seed the learner's policy with known-good demonstrations, quickly bootstrapping the learning

process without relying on any explicit programming. Importantly, this frees the task expert from the dependency of also being proficient at specifying objective functions.

It is widely accepted that for robots to achieve widespread adoption and incorporation into everyday tasks, it is critical that non-experts be capable of imparting knowledge to them through familiar means. Learning from Demonstration provides a promising pathway towards this end.

1.1.1 Learning from Demonstration

Learning from Demonstration (LfD) [15] has established itself as a valuable point of interest within the human-robot interaction (HRI) research community, particularly with respect to the intuitive interface it provides users that have never interacted with robots. Members of the HRI community have performed user studies on various LfD techniques, and continue to present results valuable to interaction designers [16–19].

In [6], a semi-supervised method of acquiring a reward signal from humans is presented, combining Relative Entropy Policy Search [20] and Bayesian Optimization within a reinforcement learning context to learn effective motor primitive policies when given the ability to occasionally query a human expert. In [21], Doerr et al. present an algorithm that enables inverse optimal control techniques to be applied within high dimensional problems, using a form of Structured Prediction to facilitate policy search. To mitigate the unreasonable requirement of having large numbers of demonstrations during training, Kim and Pineau introduce Maximum Mean Discrepancy Imitation Learning (MMD-IL) [22]. They present an iterative policy learning approach that learns a family of policies, with each individual policy learned through MMD-IL specializing in particular regions of the problem space (where others made mistakes).

Beyond policy search, demonstration-based learning techniques have been used to teach robots about objects in the world to improve their scene understanding and ability to operate in complex environments. In [23], Thomaz and Cakmak examine methods of incorporating human instruction into affordance learning for a robot. As the robot seeks to learn about the objects in its world, rare affordances are (by definition) unlikely to be discovered using uninformed, sampling based exploration methods. By leveraging human instruction ten-

dencies, structured examples can be provided and requested so as to achieve better overall task performance. Relatedly, in [24] Pillai et al. present a method that allows a robot to acquire kinematic models of dynamic objects that is informed by human demonstration. This allows a robot to better understand the ways that particular objects can be manipulated or used, extending the robot's utility and ability to plan for complex interactions in the world. Demonstrations are not limited to the physical world, as Alexandrova et al. present a means of skill learning from a single live demonstration, with subsequent information provided through simulated visualizations of the action under novel conditions [25].

Some systems have been developed to navigate the spectrum between guidance-based learning (such as LfD) and exploration-based learning (like reinforcement learning, or RL) to operate more efficiently in environments where human collaboration is expected. Systems that leverage socially guided exploration are equipped to learn from non-experts during task execution through familiar forms of social learning. Including a human in the loop for skill acquisition greatly increases the potential value of time spent training. One technique that particularly takes advantage of humans in the loop is the combination of socially guided machine learning and active learning [26, 27]. It is important to note that results from applications of these works demonstrate that the effectiveness of each training method is dependent upon factors inherent to the nature of the skill being taught as well as the comfort of the user with manipulating the robot itself.

1.1.1.1 Complex Motor Control Learning from Demonstration

Throughout this thesis, I make mention of motor primitives that are either considered to be temporally abstracted, multi-step actions (*options* [9]) within the Semi-Markov Decision Process framework or as continuously executing controllers. In both cases, LfD techniques can be used to bolster self-directed reinforcement learning algorithms, using observed or teleoperated demonstrations as seed data for policy acquisition. It is nearly always the case however that traditional, isolated policy learning techniques are inapplicable to the problem of learning such controllers in real-world scale domains. In particular, Inverse Reinforcement Learning [2, 28] provides a mechanism to utilize expert demonstrations to quickly recover an unknown reward function that may be too difficult to manually specify. With this richer

reward function available, techniques such as Q-Learning [29], SARSA [30], and LSPI [31] can be relied upon to achieve acceptable performance.

1.1.2 Active Learning

As collaborative work occurs in very large problem domains with uncertainty and unobservable teammate characteristics, the ability for an agent to help direct the course of instruction being provided to it can bring substantial benefits. Accordingly, *active learning* is a method designed to accelerate time to skill acquisition via active participation in the learning process. By enabling the learner to ask questions of its teachers, the learner can guide instructors to fill the most critical gaps in its knowledge with training data. For instance, in work by Cakmak and Lopes [32], an algorithm is presented that is capable of identifying the most desirable areas of training data within sequential decision problems. This result is applied in a user study, showing gains in performance by agents trained by humans who leveraged the agent's capacity for active learning over those that did not. Designing active learning into systems that interact with non-experts requires special care, as designing for certain query types has been shown to influence user perception [33]. Active learning remains an open topic of interest within HRI, as it incorporates a human oracle into the learning process. Guidelines for indicating appropriate types of agent queries within various types of scenarios are only beginning to emerge, and will become increasingly important as the complexity of collaborative systems increases. In addition to looking into methods that allow robots to make informed queries to others, recent work in active learning within HRI has also focused on improving the answers humans provide [34]. In this thesis, I utilize active learning to accelerate task comprehension in the context of building useful subtask abstractions.

1.1.3 Task Representation and Shared Mental Models

In addition to the training of primitive skills, enabling non-experts to develop complex interaction behavior for robots is necessary for widespread adoption. Beyond making flexible skill acquisition accessible, compiling higher-level sequences of actions must also be approachable by non-technical users. For robots working collaboratively and sharing goals

with humans, having relatable mental representations of skills and tasks contributes to behaviors that are more comprehensible, facilitating the completion of complex tasks. Internal task decomposition is at the core of this mental model, as maintaining and converging upon common mental representations of tasks facilitates interaction design across all levels of abstraction.

Fortunately, various existing algorithms are capable of decomposing tasks into representations agreeable for the internal processing of an autonomous agent [35–40]. Collaborating with humans introduces a preference for representations that are inherently comprehensible to people, avoiding the introduction of any unnecessary cognitive load when properly implemented [41, 42]. Work in this area has yielded effective results, including an action segmentation algorithm by Shim and Thomaz, inspired by human-like methods of segmentation [43] and a subgoal recognition algorithm by Menache et al. that utilizes the topology of a task's graphical representation [44]. By utilizing statistical regularities in action sequences performed by humans, a robot observer can discern meaningful options from identifying boundaries between probable groupings of actions.

Hierarchical representations, particularly Hierarchical Task Networks (HTNs), have been developed to manage the exponential complexity growth of realistic planning problems [45, 46]. This highly effective approach consists of identifying abstractions present within collections of primitive skills or subtasks to reduce planning difficulty [47]. Primitives are typically represented in a manner reminiscent of STRIPS [48], containing symbolic representations of environmental preconditions and effects.

In addition to fully automated methods, a variety of approaches have been studied involving interfaces through which non-programmers can impart plans for fulfilling complex tasks to robots. Recently, Mohseni-Kabir et al. introduced an interactive method by which a human instructor can use a software interface to teach a robot a complex hierarchical task network through a single demonstration [49]. Glas et al. evaluate novice-accessible interfaces by way of a user study in which interaction designers were able to use a graphical interface to achieve greater success in programming a complex interaction than groups that did not have access to the interface [50]. Nikolaidis and Shah demonstrate that utilizing the practice of cross-training helps to establish shared mental models; converging a robot's ac-

tion policy to user demonstrations while simultaneously informing the interaction partner's expectations about robot behaviors [5]. This involved participants swapping roles with the robot, performing its task through a simulated environment on a computer.

Methods that leverage natural language interpretation have also found success within facilitating task understanding and skill acquisition.

1.1.4 Interactive Policy Learning

Even with state spaces that can be factored through the use of hierarchical task representations, learning action policies for real-world tasks can be extremely difficult. As a result, many methods have been introduced through which humans can actively participate in the learning process, helping agents to converge on successful policies for completing various tasks. As natural language is an extremely intuitive medium for expressing direction and conferring advice, it is perhaps unsurprising that a multitude of approaches have sought to leverage this communication channel as a strong heuristic during policy acquisition.

Recently, MacGlashan et al. [7] introduced a method for grounding English commands to produce factored reward functions to use during policy learning. Using an extension of the Markov Decision Process formalism that includes object class-based abstraction (OO-MDPs), the authors incorporate a model of language grounding to provide semantic meanings to parameterizable propositional functions, inferring goal regions of state space from natural language. The contributed algorithm cleverly selects a probable goal specification by comparing descriptions of generated candidate tasks with the language used in the original task specification command. After grounding the language in a task (specified as a conjunction of planning predicates), standard policy learning techniques can then be used to solve the resulting grounded task. Rybski et al. [51] introduce an interface by which a human can train a robot verbally through the correlation of location data, agents present, and spoken commands to known skills. The work primarily leverages spoken language to allow non-experts to impart skill sequences to a mobile robot. Breazeal et al. [52] have performed studies utilizing an interface combining verbal communication with joint intention theory. Their system approaches this problem using dynamically interleaved sub-plans to produce coherent teamwork between involved parties, focusing on task representation

in terms of goals rather than solely motion trajectories. Briggs and Scheutz [53] integrate adverbial cues into natural language interpretation for building and maintaining models of collaborators' beliefs and intentions. Through understanding the pragmatic implications of adverbial modifiers (e.g., 'yet', 'already', 'now', etc.), a robot is able to update and synchronize its model of others' beliefs and intentions by accurately parsing statements, questions, commands, and acknowledgments.

Interpreting commands and verbalizing internal state are closely related, important capabilities for a robot collaborator to leverage. The policy learning process can be accelerated merely by providing an instructor with a more accurate representation of one's internal state, as this emphasizes current levels of understanding. A collaborator that is able to communicate its internal state can take steps to minimize or eliminate uncertainty through interactions with teammates. Within the realm of human-centered scene understanding, Walter et al. leverage human feedback to provide natural language descriptions usable as semantic maps, enabling robots to infer rich environmental models from the synthesis of high-level labels with low-level sensor-based observations [54]. Recent work by Misra et al. [55] presents a method that utilizes conditional random fields to ground free-form language instructions into an environment which can be distilled into a sequence of actions that completes the given task. Gemignani et al. [56] use natural language as a means of conveying a robot's unknown internal representation of the environment to human collaborators. As such, they present a method that allows a robot to warn users when it has uncertainty about object properties, in addition to learning about these properties over repeated environmental interactions. A template-based approach to modeling complex language requests that incorporates conditionals, conjunctions, and disjunctions has been used to give robots that interact with humans the ability to acquire deeper tasks strictly from spoken language [57]. This approach allows for common operators such as *AND*, *OR*, *THEN*, and *IF* to be parsed within service task specifications. Raman et al. [58] introduce formal methods into task understanding from natural language, converting natural language specifications into Linear Temporal Logic formulas. By generating correct-by-construction controllers and combining them with feedback specifying unachievable goals, an iterative task specification approach allows for the avoidance of failure modes during operation.

In addition to learning tasks from natural language, the inverse problem has also been investigated, wherein robots impart task understanding to human collaborators. Sauppé and Mutlu [59] provide insight into instructional strategies through an HRI study concerning an assembly task, showing that grouping instructions together rather than providing higher level summaries or providing individual sequential instructions contributed to faster task completion times. Their results indicate that there is a tradeoff between instruction grouping and breakdowns in understanding, and that participants were likely to fetch all prerequisite parts at once for grouped instructions, rather than sequentially.

Language has been incorporated into collaborative robots within a generative capacity for other purposes as well, allowing robots the ability to solicit others' help in overcoming limitations or to decrease others' uncertainty within a partially observable task. An extension to Partially Observable Markov Decision Processes (POMDPs), Human Observation Provider-POMDPs (HOP-POMDPs), are one formalism utilized to achieve these ends [60]. In this approach, language is used both as a means of requesting others to perform actions that achieve the robot's goals that lie outside its capabilities (e.g., "Can you please press the 'up' button?") and as a means of requesting information to reduce its sensing uncertainty (e.g., "Is this the second floor?"). Notably, this approach has been implemented on the Carnegie Mellon CoBot platform [61], a deployed system that has been performing tasks in human-populated environments and interacting with people for years. Tellex et al. [62] do so through the introduction of inverse semantics to generate useful requests for help, allowing for robots to utilize human teammates as a means of recovering from detected failures. Their approach moves away from template-based systems, and takes into account task context and scene contents, modeling the listener and the environment to select the request that most succinctly and adequately disambiguates the desired remedy from other possible solutions.

Despite this multitude of successes and clear progress, policy learning still suffers from substantial difficulties in multi-agent, human-in-the-loop, collaborative domains with uncertainty. Notably, POMDPs for decentralized agents (Dec-POMDPs [63]) do not scale to many HRI problems. In cooperative scenarios, multiple agents are typically acting in concert to achieve common goals. Communication between agents is often the only way

to synchronize via accessing unobservable information (such as future plans or goals), and may take both time and resources to perform. Therefore, decentralized planners must evaluate what agents should do between these synchronizations, despite having incomplete information to work with.

It is known that solving these kinds of problems is nondeterministic exponential time-complete (NEXP-Complete) [63]. Practically, this means that there are no polynomial-time algorithms to solve Dec-POMDPs, which imposes strict limitations on the allowable state and action space if general methods are to be utilized in solving them. In HRI-scale problems, there tend to be large amounts of possible actions, dynamic or complex environments, and tasks are typically composed of multiple steps. In practice, this necessitates the introduction of tools to facilitate abstractions and state space factorizations, limiting the size of the Dec-POMDP that is relevant at any given point in the task.

While the recently developed MacDec-POMDP [8] formalism offers a promising basis for solving these types of problems, various domain-specific heuristics must still be discovered and employed to make the learning and planning problems that exist at the scale of real-world human-robot collaborations tractable. In the following section, I outline some of these heuristics and describe some of the most relevant challenges inherent to achieving fluent human-robot teaming.

1.2 Major Challenges in Human-Robot Collaboration

In this section, I outline four major challenges inherent to building autonomous, socially collaborative systems, discuss how the state-of-the-art from the fields of HRI, LfD, hierarchical learning, and reinforcement learning can be synthesized to help solve these problems, and describe how work within this thesis addresses them.

1.2.1 Important Problems in Shared-Environment Collaboration

While acquiring skills and engaging non-technical users are clearly essential and relevant fields of research, they alone are not sufficient. The ability to collaborate is reliant upon various physical, perceptual, and social capabilities. For example, a successful teammate

often requires a model of its collaborators and the ability to communicate with them. A competent worker requires the ability to execute skills while adapting to a dynamic environment changing outside of its control. Basic operational safety demands a robot with access to a variety of sensors and real-time interpretations of the resulting data inflow, along with knowledge of appropriate actions to respond in kind. Additional software-level precursors such as emotional state recognition, object permanence, and self-evaluation of performance also dictate the potential quality of any possible team activities.

Even if a robot was constructed with dexterous capabilities exceeding that of a typical adult human, a sensor suite capable of producing features useful for complex object recognition, and hardware powerful enough to do real-time complex modeling, shared-environment collaboration in the real world presents a challenging set of research questions. Because of the great technical difficulty and hardware requirements involved, the vast majority of innovations surrounding these questions have only begun to be accomplished by the collaborative robotics (co-robots) and HRI communities.

In this section, I discuss four important research questions facing those that seek to build complete systems for human-robot collaboration. The challenges I feel to be most important within this domain are:

- Enabling and facilitating bi-directional intent recognition
- Roles selection when working with human teammates
- Evaluating trade-offs of task execution optimality against co-worker preferences
- Self-evaluation within live collaborative settings

I seek to motivate investigation into these questions with representative examples from the domains of assembly and cooking. Though many human-robot teams will comprise a many-to-many relationship, I focus on one-to-one teams as a scalable baseline for highlighting fundamental sub-questions that need be addressed within each of these four major areas.

From the domain of construction, I use the sequential manipulation task of assembling an IKEA Lätt Chair (Figure 1.1b) as a representative example. The assembly of this structure

(a) Component parts of a Lätt Chair from IKEA. (b) A fully assembled Lätt Chair.

Figure 1.1: An IKEA Lätt chair, used as a representative assembly task throughout this thesis.

requires two frames (front and rear), two side supports (left and right), and a seat to be joined together. The frames and side support components are fastened to each other with press-fit wooden pegs and metal nuts and bolts, while the seat is slid into a pre-cut channel, requiring some amount of fine motor control.

From the domain of cooking, I use the common task of baking chocolate chip cookies as a representative example [64]. The act of making cookies is a parallelizable activity that requires a chef to manipulate a multitude of materials with widely varying properties (e.g., flour, sugar, chocolate chips, etc.) and to utilize special tooling (e.g., mixers, scoops, ovens, etc.) with a diversity of affordances. This task imposes the requirement to make precise measurements and observations, to manipulate materials with delicate or complex physical properties, and to satisfy rigid temporal constraints. Given the complexity and variety of sub-tasks, many cooking tasks inherently have steps best suited to humans and steps best suited to robots.

1.2.2 Achieving Bi-Directional Intention Recognition between Humans and Robots

Humans tend to communicate intent through a variety of methods during collaborations, enabling co-workers to predict each others' future actions and plan around them. This communication can occur both verbally and non-verbally, potentially spanning multiple levels of abstraction. Teams that communicate implicitly improve performance under stresses caused by temporal constraints or uncertainty by acting in anticipation of teammates' ac-

tions [65]. For example, suppose two humans (X and Y) are constructing the chair from the assembly example above. If X requires use of the screwdriver, X's gaze may orient towards the desired object. Y may be capable of anticipating X's need based on prior observed action history, or may require more information to realize that assistance is being requested. Y can actively calibrate his intention recognition by holding the perceived object of desire in the air while focusing attention on X, as X is also likely to provide a response confirming or disconfirming Y's interpretation. This interaction communicates the same information across multiple channels, including joint attention, object manipulation, and socially meaningful gestures.

Collaborative robots must have robust classifiers capable of determining human intent, especially in situations where multiple agents' skills require coordination, such as during object transfer. By conveying intent at a high level, such as that of role selection, a co-robot can guide human teammates to choose roles with non-conflicting subtasks for greater overall efficiency. At a finer level, such as conveying intention within individual skills, the ability to broadcast one's goals or intended motion paths will lead to fewer instances of turn-taking behavior and conflicts over occupancy of shared spaces [66–68]. Intention recognition is a dynamically calibrating process, where each agent plays an active role of synchronizing communication channels and establishing expectations.

Identifying intent is especially important for potentially dangerous tasks, such as removing a hot baking tray from an oven. Tasks that may endanger either humans or robots necessitate the prior conveyance of intention for basic operational safety, beyond any safeguards inherent to the motor control sequence itself. Perceptions of danger are as important to consider as any actual danger during robot operation. Beyond safety concerns, anticipation of where objects will be placed or moved while in use directly affects the possible actions or available space usage of collaborators. As available space is a precious resource within shared-space collaborative task execution, the importance of communicating one's intent grows inversely proportionally to the amount of available working space usable by the team. In the context of the cooking example, a situation where the active working space is presumably a small area such as a kitchen, communicating one's intended motion path is critically important. Fragile or loose materials that may become dangerous under

improper handling (e.g., hot liquids) must be transported with care. This requirement can be satisfied without sacrificing advantages afforded by the parallel nature of the task, but only if team communication is effective enough to enable it.

Current research in facilitating bi-directional intention recognition has focused on pre-execution communication and predictable motion. Existing approaches rely upon attempting to characterize significant features used in human-to-human collaborative activities, such as handover tasks [69, 70]. These studies measured the importance of various features on detecting an intent to handoff an object, including partner orientation, gaze direction, hand occupancy, social gestures, and partner distance. These results were used to build a classifier with human-interpretable rules capable of recognizing handover intention within their coded dataset. Active engagement within intention recognition is a powerful tool for a robot to leverage, with the potential to greatly enhance the speed and quality of learned collaboration [71]. For example, in [72] the authors provide an analysis of gesture as a method of communicating intent from robots to humans, producing and validating a gestural lexicon within a collaborative domain. Their experimental results show that human-derived gestures can be implemented and properly interpreted when executed by a robot (in this case, a 7-DOF Barrett WAM arm), providing a mechanism by which a robot can communicate future intent or need to a collaborator.

At the level of motion planning and primitive skill execution, Dragan et al. have shown impressive results in facilitating intention projection [73,74] and improving team fluency [68] by introducing novel optimization criteria for goal- and scene-aware robot motion planners. In [75], the authors rigorously evaluate a mobile robot assistant in a human factors study. The experiment evaluated a robot that performs fetch-and-deliver tasks for a human coworker, characterizing the complexities inherent to producing collaborators that can achieve similar performance benchmarks to human teammates (in terms of interaction and idle times). Importantly, this work demonstrates evidence that robot salience is not a dominating factor in achieving fluent teaming within interactions in large environments.

In this thesis, bi-directional intention recognition plays an important part in role assignment and task understanding. In Section 3.4, I detail a method that leverages this intention recognition to allow a robot to more effectively choose roles for itself and to teach tasks to

novices.

1.2.3 Role Selection When Working with Human Teammates

When participating as a member of a team, fluency of operation can be achieved by determining a division of labor within a task and assigning the various divisions to roles. Each team member's role dictates their expected actions, simplifying the task of predicting their intent or future movement. Furthermore, a division of labor may be more efficient than single-stream task execution, especially if roles are chosen with collaborators' skills in mind. Humans are capable of decomposing tasks into multi-role, multi-collaborator endeavors, and synchronizing these decompositions between them with minimal verbal communication. This decomposition has a strong possibility of being ambiguous, requiring a wealth of contextual knowledge to disambiguate, which is unlikely to be explicitly available to a robot. Being able to generate potential role divisions from a task, synchronizing them with co-workers, and properly selecting a role based on the demands (both social and practical) of the team greatly boosts a co-robot's ability to function as an effective member of a team.

Role assignment is not a static, one-time activity. Pre-activity role determination is important for setting initial roles and expectations, but real collaboration can involve role trading and overlap, the likelihood of this increasing with task complexity. Co-workers often bridge the boundaries between roles to assist each other, even when not explicitly specified in the duties associated with an assigned role. Recognizing implicit communication and anticipating the needs or actions of co-workers contributes to fluent collaboration, increasing both the objective and perceived value of the robot [76]. Roles may change or adjust throughout a task's execution, requiring the capacity to be sensitive to others' preferences, the ability to evaluate one's own and others' skill proficiencies, the ability to evaluate temporal constraints of subtasks between and within roles, and the potential to build action policies within given safety constraints. These components must all become part of the role selection mechanism, evolving over time as more about teammate preferences are revealed.

For a team to achieve fluency of collaboration, learning patterns and appropriate reactions from task repetition is essential. Adaptive planning systems that are capable of

modeling and adjusting to human behaviors with social understanding are only beginning to be developed and tested in the real world. Important work yet to be addressed within this space includes the handling of unfulfilled commitments by teammates and handling execution-time variation in ordering constraints [77, 78]. A complete system capable of adapting to team preferences and behaviors requires many other components. In addition to the abilities listed above, a complete system requires the ability to identify roles chosen by teammates through participation and live observation, as well as the ability to estimate the impact of one's actions on others given knowledge about their goals.

Task decomposition algorithms that operate on input demonstrations have achieved success in learning execution policies from low repetitions of demonstrations of skill sequences [37,79]. They provide a great benefit through reducing the complexity of potentially intractably large problem spaces. This is accomplished by limiting the number of relevant input dimensions according to the demands of particular segments of a task. However, these task decompositions are typically optimized for internal use and are not meant to be interpreted by the user. In collaborative domains, it is important to be able to communicate one's understanding of a task in a manner comprehensible to one's teammates, while maintaining the ability to actively participate. Reconciling this notion with the output from automated task decompositions presents another aspect of the immense challenge inherent to comprehensibly representing one's mental state to another agent.

The objective of a robot teammate for any collaborative activity should be to reduce the workload, either physically or cognitively, of fellow teammates. A co-robot should be capable of identifying divisions of responsibilities within tasks, synchronizing its plan with its teammates, and self-selecting appropriate roles to assume during execution. These behaviors must be adaptive and flexible to both plan and role-assignment changes during execution. Promising initial work exists in this domain, incorporating verbal feedback to allocate and declare roles as well as synchronize task understanding during a collaborative task [80, 81].

In this thesis, I introduce a method of achieving deep task comprehension in Section 2.1. This method facilitates the learning and execution of supportive behaviors, actions that allow a robot to reduce the difficulty of collaborators' roles. Understanding the preferences

of one's teammates is crucial to establishing team fluency, and is a topic I cover in Section 3.3.

1.2.4 Evaluating Trade-offs Between Task Execution Optimality and Co-worker Preferences

When operating in mixed human/robot teams, it is inevitable that tasks will require the use of skills that certain teammates are better at than others. Likewise, there will be situations in which certain subtasks should be handled explicitly by a robot and situations where subtasks are best suited to human execution. For example, dangerous or repetitive tasks, such as cutting wooden boards or forming 1000 balls of cookie dough, are good candidates for robotic execution, while tasks that require human-human communication or high-level decision making, such as evaluating decór choices for furniture or evaluating the taste of a cookie, are ideal for human execution. Evaluating co-worker proficiency and recognizing these situations presents a difficult research challenge [82]. Even if a system were capable of this, however, the question of when and how this information should be leveraged remains.

One approach is to develop a plan over several iterations in concert with the robot teammate(s) to be used. This allows the plan to evolve naturally with all collaborators present in the process, engendering trust between workers [5]. In the future, collaborative robotics will need to enable robots to join existing teams without incurring the high costs associated with disrupting existing dynamics and redefining roles. This question introduces many others, perhaps most notably: How does one balance between preferences of co-workers, the necessities of the assigned task, and the performance criteria with which the group is being judged?

In work by Huang et al. [83], a tradeoff between performance and preference is observed within a collaborative pick-and-place task involving handovers. When comparing proactive and reactive coordination methods, users were found to elicit a preference for more reactive behaviors while team performance was most improved by proactive robot actions. In response to these findings, the authors introduce an adaptive approach that converges upon an acceptable middle ground that preserves the majority of benefits within both user experience and team performance metrics. In contrast, work by Gombolay et al. [13] found

that human workers were willing to cede decision-making authority within a scheduling role to a robot when it resulted in higher team efficiency.

Further adding to the difficulty of this question, the performance criteria may be unknown, varying by team and task. An agent must be able to successfully balance time, monetary cost, energy expended, and even emotional costs incurred by someone doing a task against their preferences. Psychological considerations may also be taken into account, including methods of handling co-worker disengagement or dissatisfaction. This evaluation function may also be subject to an external authority dictating constraints, which may occasionally be in conflict with teammate preferences. A complete collaborative system should be able to determine which trade-offs are most important and what is the best possible decision considering the team as a whole. This not only requires robust models of each teammate's intent as covered previously, but also the ability to take measurements of teammate performance, each of which evolves over time.

As a participant in a furniture assembly domain, a robot may be more proficient at uniformly and precisely applying a wood stain or adhesive compound than humans, but whether that role is a desirable choice may depend on the preferences of the human. Within the cooking domain, it may be optimal for the robot to perform steps that have strict monitoring requirements, but a human may not trust the robot to perform the task safely or with proficiency. Some tasks may be too complex or costly to perform full trials of for demonstrative purposes. How can a robot instill trust in co-workers without the luxury of demonstration as proof?

Work by Salem et al. [84] investigates the effects of errors and social factors on trust between humans and their robot collaborators. Their results showed that while humans were generally willing to obey the requests of a robot exhibiting faulty behaviors, erratic or unexpected behavior strongly influenced subjective perceptions of the robot's reliability and trustworthiness. Participants were perhaps unsurprisingly more willing to follow instructions from a faulty robot that were reversible than those that were irreversible. This suggests that for many classes of task where failure modes are neither irreversible nor catastrophic, humans may be likely to give a robot collaborator the benefit of the doubt during task execution without proof of competence (or even in the presence of counter-evidence).

The importance of measuring trustworthiness in real-time is known due to work by Desai et al. [85], as they show the typical post-experience evaluations that are used ignore crucial factors such as inertia and don't account for biases in the data, such as those from the primacy-recency effect.

Towards quantifying this latent notion of trust in real-time, Xu and Dudek introduce OPTIMo [86], a trust inference model based on dynamic Bayesian networks. Their model can be used to develop adaptive robots that seek and execute behaviors that explicitly increase collaborator trust and efficiency during teaming scenarios. OPTIMo applies causal reasoning both to perform trustworthiness assessments based on task performance and to analyze evidential factors to estimate other agents' degree of trust in the robot. In lieu of being able to measure trust, as it is never a directly observable quantity, the authors measure OPTIMo's ability to predict trust-induced behaviors and assessments (such as user interventions and trust change reports) with considerable success. This is particularly encouraging, as it suggests that notions of trust are able to be computationally modelled and reasoned with, a critical requirement within human-robot collaboration.

In Section 3.2 I describe a Task and Motion Planning approach to learning and executing supportive behaviors. In addition to providing an algorithm for doing so, I also discuss and analyze performance considerations and teaming implications imposed by a supportive robot's action selections. I propose a variety of weighting functions that can be used to modulate robot behavior between objective performance improvements and worker preferences. In Section 3.3, I address issues of trust and reliance through transparently representing a support robot's internal policy, using a language model tied to planning predicates to both learn and explain behaviors.

1.2.5 Evaluating Performance During Live Collaboration

Interacting with multiple agents in a shared task presents opportunities to evaluate personal proficiencies. While individual skills often include measures of success that can be evaluated during execution, assessing one's proficiency at being an effective and useful team member can be less straightforward. Determining this quality requires a shared mental model of the task, knowledge of co-workers' roles and responsibilities, and accurate estimates of

expected overall task progress at given times. Even still, determining the appropriate metrics and acquiring reasonable real-time measurements presents a substantial research challenge. Once acquired, this information can be used to reinforce role selections, refine skill execution choices, and evaluate novel task decompositions.

A collaborative robot must use some form of contingency detection, the detection of a change in an agent's behavior within specified time bounds of another agent's action, to measure the impact it has on its teammates at small timescales. This phenomenon has been studied within HRI for several reasons, including its utility in understanding differences between oneself and others [87] and in general for classifying responses (or non-responses) to actions [88]. Some planners have begun to use these behavioral or temporal fluctuations to increase the fluidity of team operation, adapting to perceived co-worker preferences in task execution [89].

Individuals are capable of using a wide variety of metrics to gauge success. Beyond mere evaluation of whether the group completed the assigned task, self-evaluation can include questions measuring personal performance or growth as well. A co-robot must be able to judge whether it performed its assigned tasks properly; for instance, it must be able to evaluate not just whether it has made chocolate chip cookies, but whether the cookies actually taste good. Success may also partially be defined as self-improvement with regard to various sub-skills used throughout the task; for example, a robot may have developed its insulation cutting skill to be more efficient during construction. Building cohesive teams requires pairing workers who value each others' contributions, so co-worker satisfaction and satisfaction levels of authority figures should also comprise a component of a comprehensive self-evaluation function.

Especially during the execution of autonomously generated supportive behaviors (Section 3.2), it is necessary for a robot to evaluate the impact that its actions will have both on the task being completed and its coworkers. When performing policy learning in the context of providing assistance, this self-evaluation becomes the reward signal for determining which behaviors are given preference.

(a) The KUKA YouBot mobile manipulator.

(b) Rethink Robotics' Baxter platform.

Figure 1.2: Robot platforms used within the collaborative task execution research in this thesis.

1.3 Robot Platforms

Two types of robot were used throughout the work described in this thesis: the KUKA YouBot arm (Figure 1.2a) and Rethink Robotics' Baxter (Figure 1.2b).

The KUKA YouBot arm, made commercially available worldwide in 2011, is a lightweight industrial manipulator with 5 degrees of freedom. The YouBot is .65m tall, with a work envelope of approximately $0.5m^3$, a maximum payload of 0.5kg, and 0.1mm repeatability. This platform is used for its similarity to large-scale industrial robots, modular gripper interface, and ability to operate on a tabletop surface that would be shared with human workers.

Rethink Robotics' Baxter platform is one of the first commercially available robot platforms to be marketed as 'human-safe', having been designed for the particular use case of working alongside humans. Baxter is a 1.905m tall upper-torso humanoid robot weighing 306 lbs. It has two manipulator arms, each with 7 degrees of freedom and 1.2m of reach (and 0.76m of gantry position reach). Each arm is capable of lifting a 2.2kg payload, with 4.4kg of gripping torque. Baxter has an RGB camera enclosed within the wrist of each arm, as well as in a tablet that doubles as a facial display. These cameras capture images at a resolution of 1280x800 @ 30Hz. Baxter also has a sonar ring positioned around the crown

of its head and infrared distance sensors in each wrist, though neither were used for the experiments within this thesis. Baxter is well suited for collaborative human-robot work, as it has a large operational workspace and is already deployed in production environments.

1.4 Contributions

This thesis provides contributions that address the following challenges inherent to enabling human-robot collaboration:

- How can robots learn rich, legible, hierarchical task representations from humans for use in collaborative activities?
- How can robots provide assistance and support in domains where they lack either the authority or capability to operate independently?

In Chapter 2, I address the topic of building informative and useful task representations. By leveraging a hybrid approach combining the strengths of automated planning techniques and socially guided learning (with human-in-the-loop demonstrations), I present a series of algorithms that can be used to construct legible hierarchical goal networks for arbitrary sequential manipulation tasks using properties of the underlying task graph's topology. I further examine the utility of the methods used in building this abstract task representation within the context of informed exploration functions for automated task learning, showing improvements over classical random exploration techniques. This chapter provides the following contributions:

- Clique/Chain Hierarchical Task Networks (CC-HTNs): a novel type of hierarchical task network built upon the topology of task subgoals
- An algorithm for autonomously constructing CC-HTNs from arbitrary task networks
- Novel exploration algorithms for rapidly learning ordering constraints of sequential manipulation tasks with decentralized instructors

In Chapter 3, I present results enabling robots to participate in tasks where they may have been considered previously inapplicable, providing assistance and support to their

collaborators. I approach this problem from both a skill and policy learning perspective, proposing both an automated Task and Motion Planning method of synthesizing supportive behaviors as well as a method that learns from demonstration. I investigate mechanisms of policy transfer between humans and robots, presenting an approach that enables a robot to communicate its future actions and reactions via natural language to its collaborators, in addition to parsing such language to bootstrap its own policy during learning. The chapter concludes with work illustrating two applications of the supportive behaviors framework, the first involving a robot that can transition its role from student to instructor by using learned task knowledge to train humans, and the second involving a robot expert facilitating a novice agent's self-guided exploration within a task without direct intervention. This chapter contributes:

- A definition and taxonomy of supportive behaviors, actions that teammates can execute to facilitate the work of others.
- A Task and Motion Planning approach to autonomously synthesizing supportive behaviors, modeling collaborator actions to provide productive assistance.
- An approach to learning policies for supportive roles from demonstration and collaborative discourse.
- An method of using intention projection as a social signal for transitioning a robot teammate between learner, peer, and instructor roles.
- A model for facilitating safe, self-directed learning and task exploration in novice teammates, based on supportive behaviors.

Finally, in Chapter 4, I summarize the contributions of this thesis and outline future directions and applications of this work. I address the domains of inverse reinforcement learning, multi-agent planning, and complex semantics in discriminative and generative settings.

Chapter 2

Hierarchical Task Networks for Human-Robot Collaboration

Robots that are productive teammates that can efficiently support their collaborators require a deep level of task understanding, affording them the ability to make inferences, predict intentions, and react to dynamic environments where they are not fully in control. Throughout this thesis I work under the assumption that these robot collaborators are lifelong learners and as such do not exist in isolation, either in physical space or in assigned task. A critical aspect of teamwork is the establishment of a compatible mental task model between collaborators. Due to the shared environment and close-quarters collaboration inherent to human-robot teaming, it is important that the robot's developed mental models be transparent and human-interpretable.

Hierarchical Task Networks provide a means of expressing abstraction, which is important both for transfer learning and problem decomposition. Transfer learning, or applying previously acquired knowledge across new contexts, allows an agent to apply work from prior solutions to new problems. The ability to utilize previously learned structure in new problems allows for the distribution of learning cost over a wide variety of domains. Problem decomposition allows for a task to be factored into smaller parts that are each easier to solve than the whole, enabling existing planning methods to apply in complex application areas.

This chapter describes introduces Clique/Chain Hierarchical Task Networks (CC-HTNs), a cornerstone of this thesis' contributions, as they provide methods for factoring complex tasks into tractable units. In introducing CC-HTNs, I also present a novel mechanism for autonomously extracting hierarchical structure from graph-based task representations (such as Markov Decision Processes) to construct them. The chapter concludes with an analysis of using the intermediate graph structures from this method to create informed exploration methods that are more likely to discover valid task execution paths than other methods not informed by the task graph's topology.

A brief survey of relevant work from the Learning from Demonstration, Markov Decision Processes, and Hierarchical Task Networks research communities is provided to familiarize the reader with the techniques that form the basis of my approach.

2.0.0.1 Learning from Demonstration

The acquisition of skills and task solutions from novice users via demonstration, Learning from Demonstration (LfD), is widely considered to be a simple and natural method of robot training [90]. LfD is particularly useful due to its vast applicability, as demonstrations can be effectively provided via teleoperation [28], observation [1, 16, 91] and kinesthetic teaching [19,92], guided by human speech [51,93] or even crowdsourced over the Internet [94] to achieve task or skill proficiency.

Recent work has extended skill generalization within LfD, a highly desirable property, by leveraging Bayesian non-parametric models and dynamic movement primitives to extract flexible skills from unstructured demonstrations [95]. This represents one potential opportunity for applying the work presented in this chapter, providing deeper task-level structure that may not otherwise be emergent from the segmented trajectories produced by similar approaches. LfD research has also been conducted to enable robots to learn from demonstrations performed by other robots [1, 96], leveraging observations of other robotic agents to improve one's own performance.

A key benefit of LfD is that it enables novice users to easily program robot behaviors. Autonomously obtaining complete and relevant symbolic groundings for LfD-acquired skills is difficult, making such skills ineligible for proper inclusion into traditional symbolic

planners. Regardless, even if all effects or object properties relevant to a motor primitive cannot be precisely specified, these skills can still be utilized within policy-based learning techniques, a primary motivation for my work.

2.0.0.2 Markov Decision Processes

Graphical task representations are desirable for their expressivity. Markov Decision Processes (MDPs) provide a convenient and efficient framework for planning.

The MDP framework, represented by the 4-tuple (S, A, R, P), defines an environment-centric directed multigraph with environmental knowledge encoded in vertices representing possible states ($s \in S$). Directed edges between vertices (transitions) are labeled with an action ($a \in A$). $R(s'|s, a)$ is the reward achieved by transitioning to state s' from state s via action a. $P(s'|s, a)$ characterizes the dynamics of the system, indicating the probability of arriving in state s' from state s when executing action a.

In the MDP framework, agents acquire a policy π that informs action selection, producing a state-action mapping function. In Semi-Markov Decision Processes (SMDPs) [9], these actions can encapsulate arbitrary levels of complexity, typically in the form of a temporally abstracted, parameterized motor primitive. These flexible, arbitrarily complex options are defined as closed-loop policies describing action sequences [9,97]. Accordingly, the temporal abstraction and generality of representation make SMDPs a popular method of internally representing knowledge about actionable skills and tasks [14, 79, 98, 99]. Formally, an option (skill) consists of three components: a set of initiation states from which it can begin execution, a set of termination states and associated likelihoods from which it is considered completed, and a policy for execution that transitions the environment from initiation states to desirable termination states.

The generality and extensibility of this approach contributes to policy-based learning being a widely used method of skill representation [99]. With an environmental reward function, solving for an optimal action policy can often be accomplished autonomously and is only limited by the complexity of the problem's state representation. Consequently, heuristics that reduce the number of trials required to achieve competent policies are desirable, as the exploration space of an MDP scales exponentially with the dimensionality of the state

vector and action set.

2.0.0.3 Hierarchical Task Networks

HTNs scale more readily and are more expressive than STRIPS-style planners [100]. These planners have been successful in a wide variety of rich and complex problem domains, including internet-scale distributed device planning [101], multi-agent team coordination in robot soccer [102], mobile robot exploration [103], construction [104], production scheduling [105], and crisis management [106]. As such, they are highly desirable task representations that serve a multitude of important uses. In particular, AND/OR graphs [107] have emerged as a popular choice within multi-agent assembly domains [35, 108].

However, these hierarchical networks are typically assumed to be pre-specified or assume the availability of rich domain knowledge, planning operator (action) descriptions, or substantial manual intervention to facilitate their discovery and/or specification [49,109,110]. I present a novel bottom-up approach to HTN generation that does not require prior manual intervention, building on the causality analysis principles of Nehta et al. [111] and subgoal discovery of Menache et al. [44] to find and abstract flexible subtasks.

Combining user-friendly skill acquisition and robust task-level planning in real-world systems is critical to the widespread adoption of collaborative robots. While demonstration-based training techniques inherently afford ease of use, their informality comes at the expense of not necessarily producing complete specifications of their skills' intentions or environment effects [2]. On the other hand, flexible planning systems are able to operate at varying levels of abstraction to quickly produce efficient solutions [112]. These systems typically require either manually annotated hierarchical structure or well-specified preconditions and effects of their atomic-level planning components [113]. Just as humans do not rely on explicit annotations or out-of-scenario interventions, we expect competent robot partners both to learn skill execution policies and to infer hierarchical task structure from observation or experience.

In addition to producing acceptable sequences of skills, planning systems represent an agent's awareness of the task's underlying structure, allowing for adjustment in case of unexpected deviation. While robots operating in isolation may seek to minimize a cost function

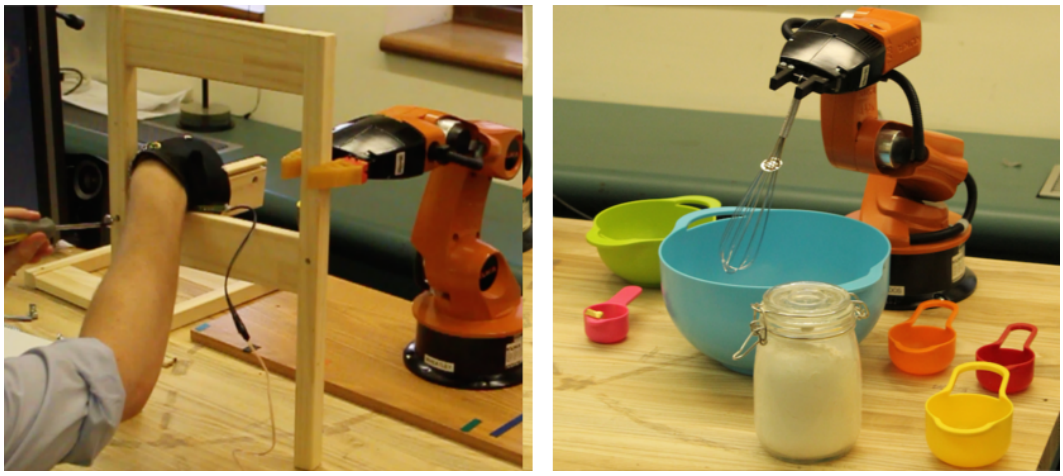

Figure 2.1: Collaboration relies on complex planning and intention recognition capabilities. For many tasks, especially within the assembly and cooking domains the HTN evaluations are based upon, hierarchical models are required to identify and solve tractable sub-problems.

specified in terms of effort or time required, it may be desirable for robot teammates to instead choose to optimize for flexible execution or maximally supportive roles [114]. As such, collaborative robots require a deeper understanding of tasks than their isolated counterparts, and must be capable of reasoning about and planning around potentially complex subtask assignment and resource management constraints [38]. Therefore a robot becomes increasingly capable of adapting to changing circumstances by deferring decisions, mitigating both its own failures and those of its partners as it learns more about the underlying structure of a task. This is known as *least-commitment planning*, an important concept in multi-agent domains with uncertainty [77].

Effectively accomplishing tasks in collaborative domains demands a high level of execution flexibility that can only emerge through task comprehension. Manually providing a robot with this kind of knowledge is difficult and time consuming and does not scale well with increasing deployment of robots in the workforce or home. Mechanisms that allow robots to build their own hierarchical interpretations of tasks are central to addressing this issue.

An ideal deployed robot would require a minimal level of training, provided by a non-expert in robotics, to reach a base level of competency for a given task. This robot should then be able to progressively learn more about the task it is performing over time, eventually

autonomously acquiring a full understanding of the relationships between the steps within the task. Once a sufficient level of understanding is acquired, the robot would then be able to work as part of a team, leveraging its acquired knowledge to facilitate collaboration and hasten comprehension of more complex tasks. Mechanisms that allow robots to build their own hierarchical interpretations of tasks are central to addressing this issue.

Tasks suited to such a robot can be found throughout commonly described robot application domains. In a factory setting, a robot could increase its utility beyond isolated assembly line tasks to work collaboratively in a shared environment to increase the productivity of a human team. Within the home, a robot could use commonalities between task hierarchies to generalize its experiences to new contexts, for example encapsulating the steps for a "chop peppers" subgoal (e.g., "Get Knife", "Cut peppers", "Place knife") from an omelette cooking task and reusing it in a pizza cooking task. Such transfer capability increases the agent's value through routine operation, speeding comprehension of new tasks.

To build collaborative, user-friendly, trainable robots, a method of autonomously deriving legible hierarchical task structure is required. In the following section, I present an approach that does so by leveraging ordering constraints and contextual knowledge from previously known tasks to learn multiple levels of abstraction for arbitrary task networks. This approach holds a minimal set of assumptions about skill representations, allowing it to be applied even to systems trained solely with skills that lack well-known preconditions or effects (or other such formalization). Additionally, the presented hierarchical representation offers transparency to collaborators, as it affords an agent the ability to summarize the internal state of its task understanding in an easily interpreted tree representation.

2.1 A Hybrid Approach to HTN Creation from Demonstrations and Planning

Collaboration between humans and robots requires solutions to an array of challenging problems, including multi-agent planning, state estimation, and goal inference. There already exist feasible solutions for many of these challenges, but they often depend upon having rich task models. In this section, I detail a novel type of Hierarchical Task Network called

a Clique/Chain HTN (CC-HTN), alongside an algorithm for autonomously constructing them from topological properties derived from graphical task representations. As the presented method relies on the structure of the task itself, this work imposes no particular type of symbolic insight into motor primitives or environmental representation, making it applicable to a wide variety of use cases critical to human-robot interaction. I then present evaluations within a multi-resolution goal inference task and a transfer learning application showing the utility of my approach.

The primary contributions of this section are:

- Clique/Chain Hierarchical Task Networks (CC-HTNs), a new type of HTN that encapsulates topological properties of a task's action space.
- An algorithm for autonomously constructing CC-HTNs from task graphs
- Evaluations of CC-HTNs within state estimation and knowledge transfer for reinforcement learning domains

2.1.1 Task Discovery Through Constraint Analysis

As the learner discovers novel paths or invalid state transitions within a task, insight into the structure of the task being performed increases. Given a graph $G = \{V, E\}$, with vertices corresponding to environment states and directed edges indicating available transitions between them, I define a task execution x as a directed graph induced by the path on G that is followed during the execution of a particular valid sequence of motor primitives. I define the set X as that consisting of all possible, efficient (having no superfluous actions) task execution paths that terminate in success (i.e., assumptions similar to [111]). Thus, $x = \{V_x, E_x\} \in X$ describes a graph formed from a single path through a set of vertices in $V_x \subseteq V$ connected via edges in $E_x \subseteq E$ that are labeled with actions $a \in A$, where A is a dictionary of known primitive skills (options) and their valid parameterizations. Learning from Demonstration provides a means to acquire these x quickly, via instructor-provided examples. Alternatively, these x can be generated by an automated planner with some degree of stochasticity or exploration. More plainly, each graph x is a chain of vertices

linked by actions that results in task completion, a path forming a subgraph in the complete task space defined by all possible states and action connections. This path originates from a vertex that is an initiation state of G and terminates in a goal state of G.

I define the task structure as being fully knowable when the learner's task graph can be reduced to a subgraph containing only the vertices and edges of successful (or in a planner's case, satisfying) task executions. Given each possible task solution $x_i \in X$, I define a task being iteratively learned by the robot as the weakly connected directed multigraph $T = \bigcup_{i=1}^{|X|} x_i$, describing an MDP-like graph consisting of all known valid execution paths. This construction is especially useful when using low-repetition, demonstration-based training common to deployed robotics scenarios.

In entirely demonstration or observation driven scenarios, my approach imposes the requirement of autonomous task failure detection, which can be difficult in LfD-driven, real-world scenarios. As a consequence of this, I qualify my approach as 'weakly supervised' for that particular domain, while remaining fully autonomous for more well specified applications such as symbolic planning.

2.1.2 Task Encoding and Action Representation

For generality, I encode tasks with the traditional SMDP representation $\{S, A, R, P\}$ which, when paired with a transition policy π, provides a straightforward execution plan for an agent to follow and optimize. In this formalism, states are uniquely described by their environmental conditions. As the description that follows remains very generalized, it may help the reader to imagine these states as being feature vectors of symbolic planner fluents (e.g., "book on table", "robot gripper empty"), though my formulation retains generality to any type of descriptive feature vector.

Due to the lack of assumptions regarding symbolic skill knowledge, I represent environment states as compositions of functions $f_{a_n} \circ f_{a_{n-1}} \circ ... \circ f_{a_1}(x)$, where x is the initiation state at the time of execution and $f_{a_i} : S \to S$ represents the effect that a_i has on the environment. This function composition serves to map environment states satisfying preconditions of a_1 to environment states satisfying postconditions of a_n. Each of these f_{a_i} is taken directly from the action sequence followed by the agent through the state space to

reach the current state. I do assume the capability to identify commutativity between f_{a_i}, as these compositions will result in equivalent final states.

2.1.3 Conjugate Task Graph

I desire a representation that reveals easily exploitable relationships between skills to help discover logical groupings of actions to abstract into subtasks, free of execution context. In providing this abstraction, I sacrifice provable optimality in favor of tractability. Environment-centric representations of SMDP-like graphs do not readily convey features facilitating the discovery of a task's underlying structure. To overcome this, I augment the task graph via a transformation function (Algorithm 1) to produce a constraint network in a compact form of its conjugate, resulting in a graphical representation with actions encoded in vertices as subgoals and environmental prerequisites encoded on its edges as compositions of subgoal completions. I refer to this graph as the *Conjugate Task Graph (CTG)* (Figure 2.2).

To create a CTG, I follow the procedure described in Algorithm 1. As input, I require a sequential manipulation problem represented as a task graph (obtainable by teleoperation, demonstration, or exploration), with options specified at the level of subgoals. Adapting arbitrary primitive actions to serve as subgoal specifications may involve utilizing a HI-MAT hierarchy [111] or learning options that achieve the 'bottleneck states' discovered via a state space segmentation algorithm like Q-Cut [44] to derive higher-level grounded options. In the IKEA assembly domain, these options are shown in Figure 2.2. Recall that the goal of this algorithm is to create a graph where subgoals are represented as vertices and environmental constraints are represented on edges.

The algorithm begins by creating the virtual states *origin_vertex* and *terminal_vertex* to act as a source and sink representing the start and end conditions for the task. Then, a labeled vertex is added for each parameterized action that appears in the SMDP (a subset of the set A in the SMDP 5-tuple). Once the vertices of the graph have been defined, the main loop of the algorithm is executed, populating the edges of the graph based on the input SMDP's state connectivity within the family of satisfying execution policies.

In lines 8-11, the initiation and goal sets are transformed into the new representation. For any task graph edge e originating at an initiation state, a corresponding edge is created

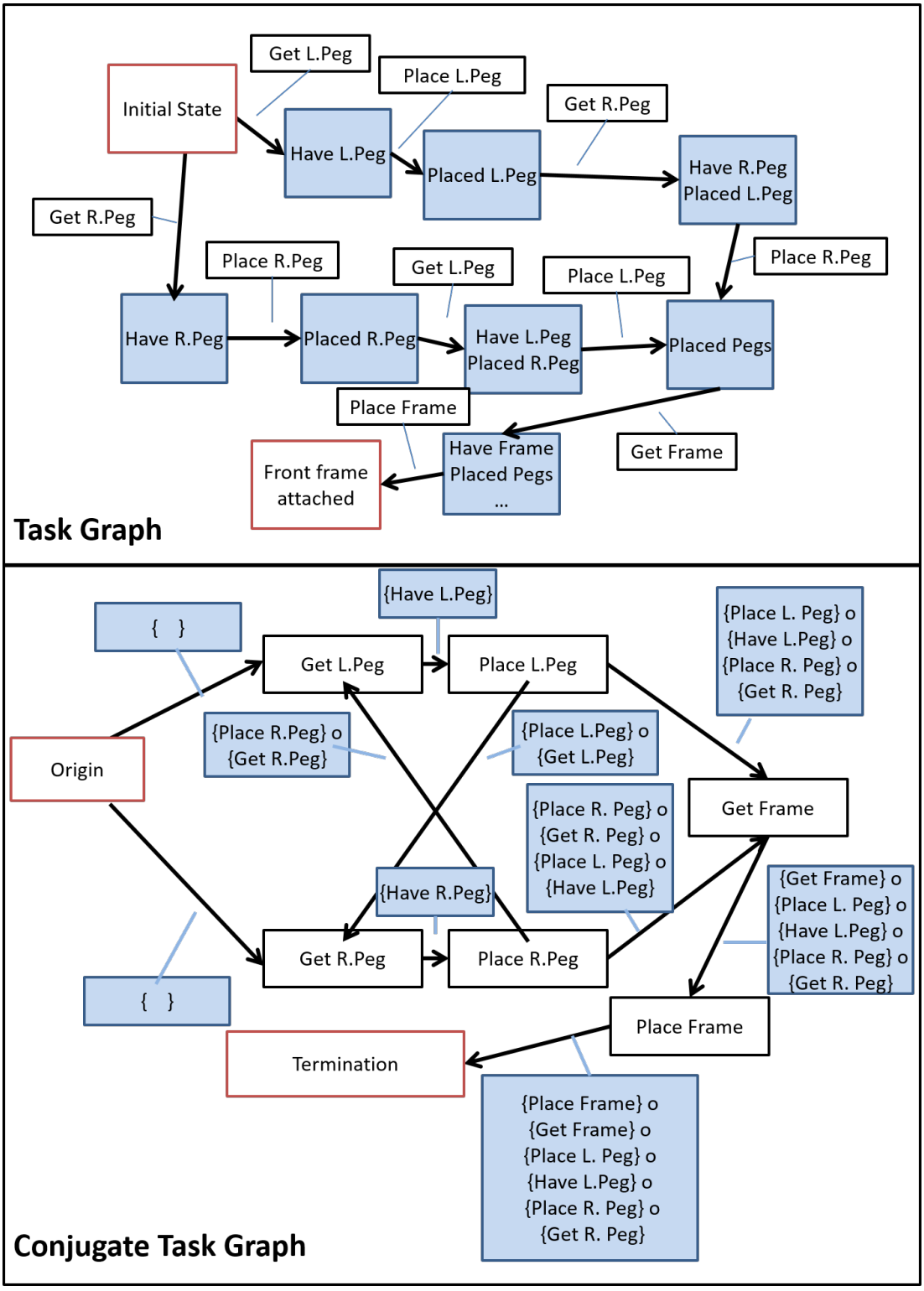

Figure 2.2: Task graph (top) and its conjugate (bottom). In the task graph, environment states are encoded in vertices and edges are labeled with their corresponding actions. In the conjugate, subgoals are represented as vertices and edges are labeled with environmental prerequisites.

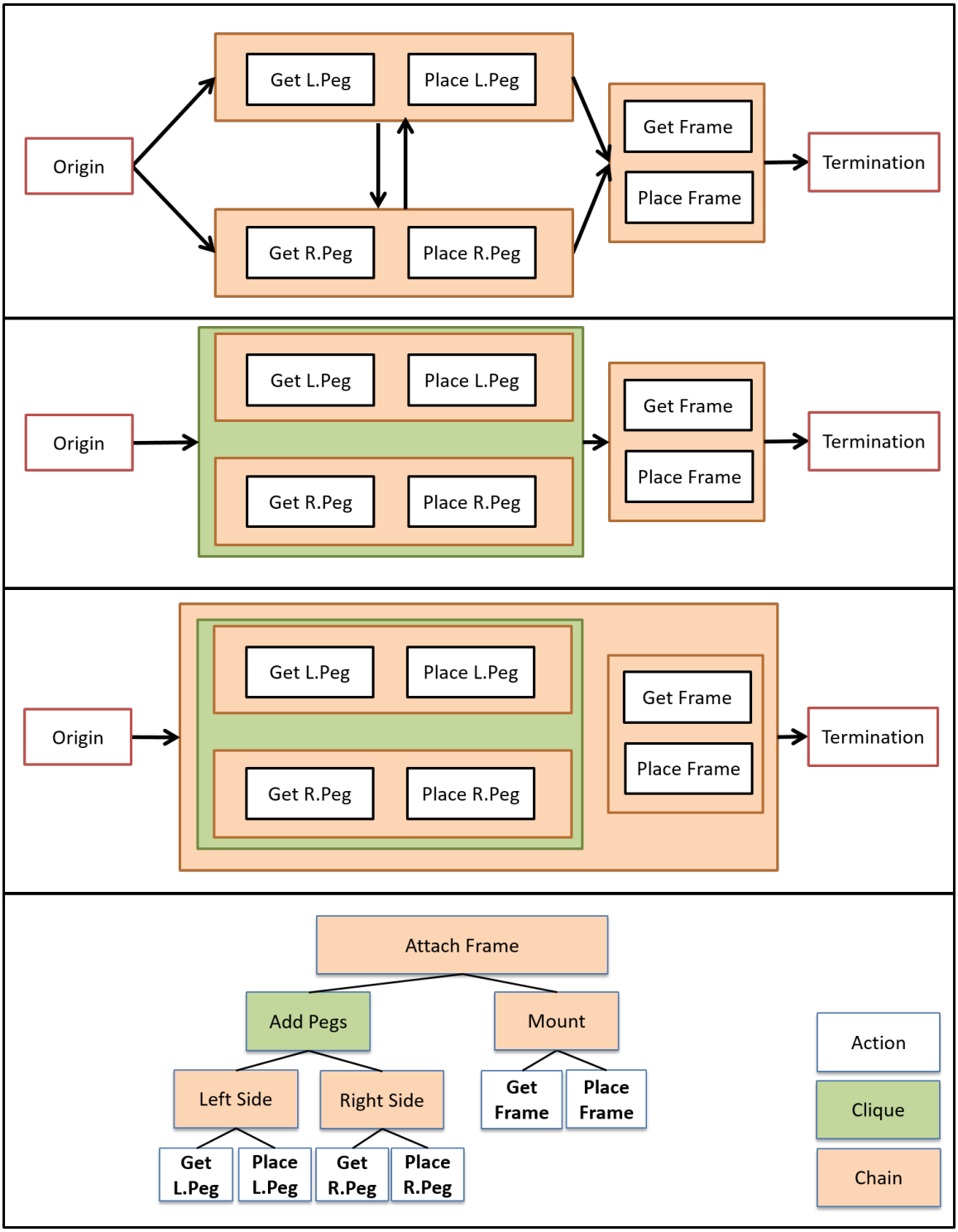

Figure 2.3: Incremental execution of Algorithm 2. For clarity, I omit edge prerequisite conditions. The internal node labels on the final HTN were manually added for clarity, illustrating the implied logic behind the action grouping.

in the conjugate graph connecting the *origin_vertex* to the vertex labeled with the action represented in e. Similarly, for any graph edge e terminating at a goal state, a corresponding edge is made connecting the conjugate vertex with the same action label to the graph's *terminal_vertex*.

In lines 13-14, the internal edges of the conjugate graph are populated by looking at pairs of edges (e, f) that share a common vertex in the original task graph. Edges are added to the conjugate graph from the vertex matching the action on e to the vertex matching the action on f. Finally, in lines 15-16 I remove redundant edges in the graph, contributing towards the HTN only enabling plans with forward justified steps [115].

Algorithm 1: Conjugate Task Graph Transform

Input: SMDP-like Graph $G = \{V, E\}$
Output: Conjugate Task Graph C

```
1  C ← empty Conjugate Task Graph {W,F};
2  origin_vertex ← new empty vertex;
3  terminal_vertex ← new empty vertex;
4  Add origin_vertex and terminal_vertex to W;
5  foreach unique a ∈ {action(e)|∀e ∈ E} do
6      Add vertex {action = a} to W;
7  foreach edge e ∈ E do
8      if source(e) ∈ initiation_states(G) then
9          Add {from: origin_vertex, to: [v ∈ W ∋ action(v) = action(e)], prerequisites:
             ∅} to F;
10     if dest(e) ∈ termination_states(G) then
11         Add {from: [v ∈ W ∋ action(v) = action(e)], to: terminal_vertex,
             prerequisites: environment_state(dest(e))} to F;
12     else
13         foreach edge f ∈ outbound_edges(dest(e)) do
14             Add {from: [vertex v ∈ W ∋ action(v) = action(e)], to:
                 [u ∈ W ∋ action(u) = action(f)], prerequisites:
                 environment_state(to(e))};
15 foreach a, b ∈ F ∋ a ≠ b do
16     if from(a) = from(b) and to(a) = to(b) and prerequisites(a) ⊆ prerequisites(b)
         then delete b;
```

Algorithm 2: Construct CC-HTN

Input: Conjugate-Task-Graph G
Output: HTN H

$H \leftarrow \text{Copy}(G)$
while $|H_{vertices}| > 1$ **do**
  $h\_size \leftarrow |H_{vertices}|$
  foreach *maximal clique* $c = \{V, E\} \in H$ **do**
    Compact $\{V \in c\}$ into single metanode m
  foreach *maximal chain* $c = \{V, E\} \in H$ **do**
    Compact $\{V \in c\}$ into single metanode m
  if $h\_size == |H_{vertices}|$ **then** break;

2.1.4 CC-HTN Generation

I now introduce the Clique/Chain Hierarchical Task Network (*CC-HTN*), a novel HTN based on subtask ordering constraints. This method draws its utility from the observation that the basis of a task network's structure is embedded in the restrictions placed on allowable permutations of subgoal sequences during execution. These restrictions can be characterized as independently applying to subsequences of goals or motor primitives in the task's hierarchy, indicating ordering constraints on a subtask. My approach to hierarchy construction classifies subgraphs into two fundamental types: order-agnostic sequences (cliques) and order-invariant sequences (chains).

Composing the CTG into collections of order-agnostic and order-invariant subsequences provides a mechanism by which a task hierarchy can be derived solely from known transitions. To finish the process of converting a task graph into a CC-HTN (Algorithm 2), I perform a series of alternating, contracting graph operations on the CTG. These operations are performed until either the task is condensed into a single vertex, indicating a successful and complete hierarchical abstraction, or until the graph does not change as a result of the actions, indicating that the graph is either incomplete or cannot be condensed further. This process is visually represented in Figure 2.3, with failure contingencies explained in the following subsection.

A *clique* in a CTG has at least one set of inbound edges such that one member edge terminates at each internal vertex of the clique. These edges must all share a common

prerequisite list on their label. There must also exist a set of edges originating within the clique, with origins at each clique member, terminating at the same external node. These edges share the requirement that each edge's label must minimally match the postcondition set resulting from the execution of all skills represented within the clique. Due to the construction of the graph, it can be inferred that the environment-modifying functions of skills within a clique are commutative with respect to successful task completion. This can be seen in the "Add Pegs" node (Figure 2.3), as the order doesn't matter when placing the left and right peg.

A *chain* is defined as a path of vertices fulfilling special connection criteria. A chain has a starting vertex with out-degree one and a termination vertex with in-degree one. All intermediate members of a chain must be on a path between the starting and ending vertex, with an in-degree and out-degree of one. This occurs in the "Mount" node (Figure 2.3), as getting then placing the frame must occur in sequence.

By iteratively finding and replacing maximal chains and maximal cliques in the CTG with meta-vertices until either a single vertex remains or the graph remains unchanged, a hierarchical structure emerges from known (or hypothesized) graph transitions (Figure 2.3). Hierarchies generated by this method have the benefit of being human interpretable (given sensibly named motor primitives).

2.1.5 Hierarchical Ambiguity

Three conditions exist in which a task does not cleanly abstract into a hierarchical structure with a single root node.

2.1.5.1 Redundancy or Equivalence

A maximally abstracted hierarchy cannot be guaranteed with my algorithm if actions tangential to the goal of the task are included. Any unnecessary actions must be identified and removed from the task graph. Similarly, if two or more skills perform equivalent functions from a goal fulfillment perspective, they must be classified as instantiations of the same skill (e.g., pressing a button with the top or side of a manipulator).

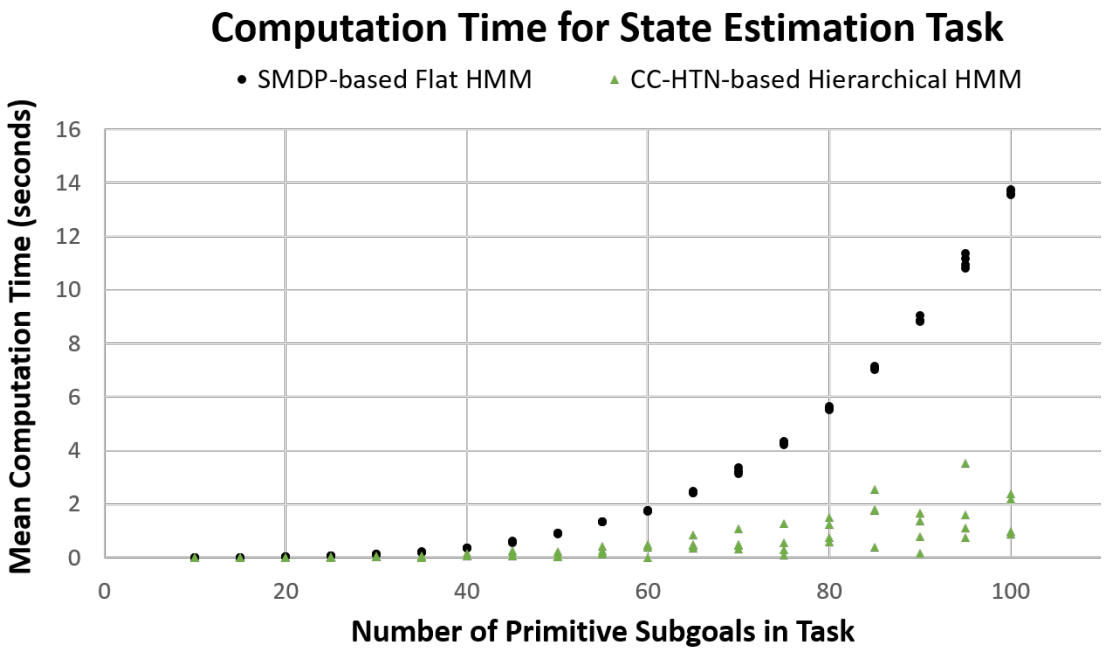

Figure 2.4: Mean computation times for estimating an agent's last completed subgoal using HMMs, as a function of task complexity. Timings were measured using a single thread of an Intel i7-3930K CPU. The responsiveness required in collaborative task execution mandates high frequency state estimation capabilities. In most cases, the CC-HTN model completed its computation faster than 1Hz.

2.1.5.2 Incomplete Graph

If the task graph does not contain all valid subtask/action ordering knowledge, only a partial hierarchy can be provided. Autonomous exploration or active learning can be used to discover these ordering constraints [116]. Algorithms 1 and 2 can be applied online, providing more structure as new demonstrations are observed and the task graph's topology is discovered.

2.1.5.3 Multiple Valid Alternatives

Many subtasks have multiple valid constraint-based hierarchical representations. One such example is the task of placing two glasses on a table then filling them. If the possible actions were {Place Glass 1, Place Glass 2, Fill Glass 1, Fill Glass 2}, it is equally plausible that this can be abstracted to {Place Glasses, Fill Glasses} or {Place/Fill Glass 1, Place/Fill Glass 2}. These situations present themselves as vertices in the CTG that are both part of

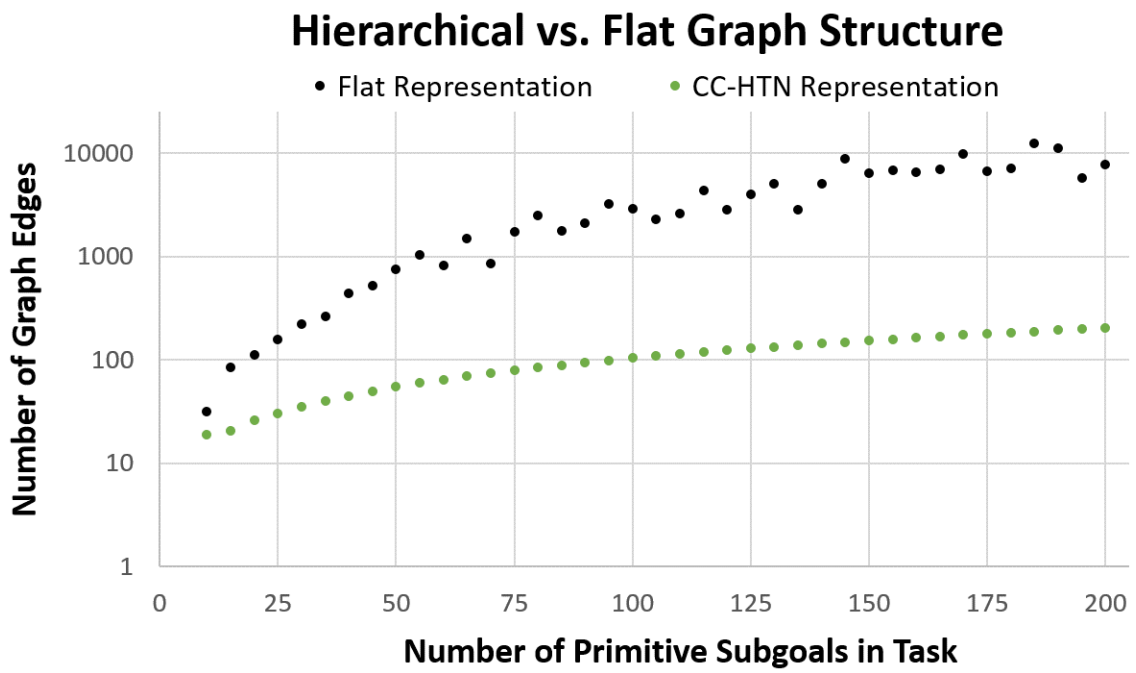

Figure 2.5: Average number of edges in generated sequential manipulation tasks as a function of subgoal count. The CC-HTN encapsulates much of the transition complexity in the task, causing a dramatic reduction in the amount of edges required to represent task substructure. This simplification of task dynamics allows for more rapid computations in role selection and intention recognition.

cliques and chains at the same level of abstraction. In practice, both hierarchies should be kept at each possible fork, with the final hierarchy decided by predicted execution policy reward (given the available agents, resources, etc.). In the event that Algorithm 2 does not converge to a single vertex, Algorithms 3 and 4 can be run afterwards to extract these alternative hierarchical representations if they exist.

2.1.6 Algorithm Runtime Analysis

The dominating complexity factor throughout these algorithms is the maximal clique detection within the Conjugate-Task-Graph to HTN Transform (Algorithm 2) and clique ambiguity resolution algorithm (4), an NP-hard problem. This step can be accomplished in time $O(d|V|3^{d/3})$ using the Bron-Kerbosch algorithm [117], where d is the degeneracy of the graph and $|V|$ is the number of graph vertices, on a simplified (undirected) CTG to find maximal clique candidates. These candidates are then kept or eliminated based upon verification that edge prerequisite conditions are met in the original CTG. The process of

Algorithm 3: Resolve Chain Ambiguity

Input: Conjugate Task Graph $G = \{V, E\}$
Output: Conjugate Task Graph $H = \{W, F\}$ or *null*

$C_G \leftarrow$ all sets of commutative subgoals in G;
$C_E \leftarrow$ all edges between subgoals in C_G;
$W \leftarrow V$;
$F \leftarrow E \setminus C_E$;
$new\_chains \leftarrow []$;
foreach *maximal chain* $c = \{X, I\} \in H$ **do**
    Compact $\{X, I\} \in c$ into single chain metanode m;
    $new\_chains.append(m)$
if $|new\_chains| == 0$ **then** return null;
foreach *Edge* $e \in C_E$ **do**
    if $to(e)$ *is the head of a chain* $c \in new\_chains$ **then**
        $e.prerequisites.append($ Members of c not in e.prerequisites$)$;
        $F = F \bigcup e$;
Remove redundant edges from F;

Algorithm 4: Resolve Clique Ambiguity

Input: Conjugate Task Graph $G = \{V, E\}$,
Edge set $Z = \{$Known subgoal transitions$\}$
Output: Conjugate Task Graph $H = \{W, F\}$ or *null*

$H \leftarrow Copy(G)$;
$J = \{X, I\} \leftarrow$ Resolve-Chain-Ambiguity(G);
$C \leftarrow \{$set of chains in $J\} \setminus \{$set of chains in $G\}$;
$Q \leftarrow$ Map $\{$v: $\{$subgoals commutative with $v \in W\}\}$;
foreach *Chain* $c \in C$ **do**
    foreach *Edge* $e \in$ *Chain* c **do**
        if *(*$to(e) \in c$ *&&* $from(e) \in c$*)* **then** $F \leftarrow F \setminus e$;
        $prerequisites(e)$ +=
        $\{g \in Q[from(e)] \mid g \notin prerequisites(e)\}$;
        if *(modified* $e \notin Z$*)* **then** return null;
        $F \leftarrow F \bigcup \{$modified $e\}$;
$new\_cliques \leftarrow []$;
foreach *maximal clique* $q = \{X, I\} \in H$ **do**
    Compact $\{X \in q\}$ into single metanode m;
    $new\_chains.append(m)$;
if $|new\_cliques| == 0$ **then** return null;

finding chains is linear in the number of vertices within the graph, while the conjugate graph transform algorithm (1) and chain ambiguity resolution algorithm (3) run in time that is polynomial in the number of observed vertices, edges, and task actions.

In practice, a large number of conjugate task graphs in the sequential manipulation domain will have low degeneracy (indicating a majority of subtasks being order-invariant) compared to the number of vertices within them (total number of subgoals), which drives down the cost of clique detection.

2.1.7 Applications and Evaluation

Two important applications of CC-HTNs are in goal inference and task planning. The challenge of interpreting another's actions to derive potential goals is critically important within human-robot collaboration. Intention recognition enables an agent to vastly expand its planning horizons and narrow its belief space about its teammates.

An agent that can rapidly re-plan task solutions to accommodate and work around the actions of its collaborators is far more valuable than one who cannot. As such, I conclude the section with an analysis of the CC-HTN's ability to accelerate task planning solutions by leveraging abstractions from prior experiences. This performance boost is obtained by employing a simple transfer of knowledge across tasks: internal nodes of CC-HTNs (each representing multiple motor primitives) are added to the agent's action dictionary, allowing for a planner to use larger building blocks in its search.

2.1.7.1 State Estimation

To showcase the effectiveness of the autonomously constructed CC-HTNs, I demonstrate their performance first within an online goal estimation task. In human-robot collaboration, there will always exist one or more agents that cannot be controlled by a central planner and cannot be assumed to regularly report their status explicitly. Thus, it is imperative that agents be able to infer each other's goals (e.g., "Agent 1 wants to attach the seat of the chair").

To perform this goal inference, I create hierarchical Hidden Markov Models that I assemble from autonomously constructed CC-HTNs (by Algorithm 2). HMM states represent

task subgoals, while state-state transition probabilities are normalized across valid transitions within the HTN. I use Semantic Event Chains [118] to define the observation sets for each action within the HTN, with a naive Bayes assumption of observation independence. Semantic Event Chains model actions as sequences of object-object and agent-object contact/non-contact events, thus the observation sets at each state in the HMM are these contact events. Combining this action representation with an HMM provides a probabilistic model for predicting the goals of an observed agent. The Semantic Event Chain action representation is conveniently independent from agent kinematics, and can be used for predicting both human and robot intent.

2.1.7.2 Task Definition

The tasks used in this evaluation were generated using encoded IKEA furniture assembly tasks, a domain previously explored in multi-agent collaborative scenarios [119], as a statistical basis for the task structure. These tasks had subtask commutativity/invariance, per-skill observation set overlap, and action set sizes sampled from distributions modeled after real assembly tasks. The chosen domain serves as a convenient basis for sampling arbitrarily complex tasks within which to demonstrate the scaling capabilities of CC-HTNs. The evaluation dataset included 76 tasks, each with 4 randomly determined execution paths. These tasks ranged in size from 10 to 100 subgoals, consisting primarily of parameterized pick, place, and fasten behaviors. Within the CC-HTNs, the average depth for a motor primitive ranged from 1.99 to 3.61, with a mean across all tasks of 2.84.

2.1.7.3 Results

I compare the state estimation performance of CC-HTN-based hierarchical HMMs against traditional HMMs built from the same task. This provides a helpful measure of utility for the generated hierarchies, namely how well they truly segment and abstract the problem space. I evaluated performance across two measures: state estimation computation time and graph size.

Collaborative task execution imposes strong real-time computation constraints. Hierarchical structures mitigate the growth of computational time requirements with task size. If

the generated hierarchies were mostly flat or did not segment the task well, I would expect the hierarchical HMM and flat HMM performance curves to be similar. As the computational performance results show (Figure 2.4),CC-HTNs provide considerable computational benefits over flat task models despite being built from the same base representation. These outcomes strongly imply a useful set of abstractions was created. In practice, hierarchical state estimation accuracy benefits will be dependent on observation set overlap between subtasks, however for cases where there is suitable distinction, these observed computational benefits apply.

I also evaluate the complexity of the hierarchical representation against that of the flat task graph (Figure 2.5). The primary computational advantage of my approach is derived from the simplified structural representation afforded by encoding ordering constraints within parent vertices in the generated HTN. In particular for clique substructures, this reduces the number of edges required to represent subgoal relationships from being polynomial in the number of involved subgoals ($|E| = |V|^2 - |V|$) to linear ($|E| = |V|$). For each incoming and outgoing clique transition, edge count is reduced from linear in the number of clique members ($|E| = |V|$) to constant ($|E| = 1$).

Having a hierarchical task representation allows for state estimation at multiple resolutions. In the case of assembling an IKEA chair, a predicted internal node of the hierarchical structure may represent the goal of "attach rear left leg to chair", encompassing all of its child primitives, providing broader context than merely identifying an agent attempting to "get wooden peg". Being able to identify higher level goals provides essential context when providing assistance or planning one's own actions. In many cases it is more important to recognize these goals than the motor primitives themselves.

2.1.7.4 Task Planning and Transfer Learning

In addition to policy estimation and intention prediction, the presented approach has utility within task planning, allowing an agent to leverage prior experiences (as macro actions) to learn more quickly in new contexts.

The ability to plan with high level actions acquired through normal operation is tremendously valuable for a collaborative robot. These meta-actions allow a task planner to take

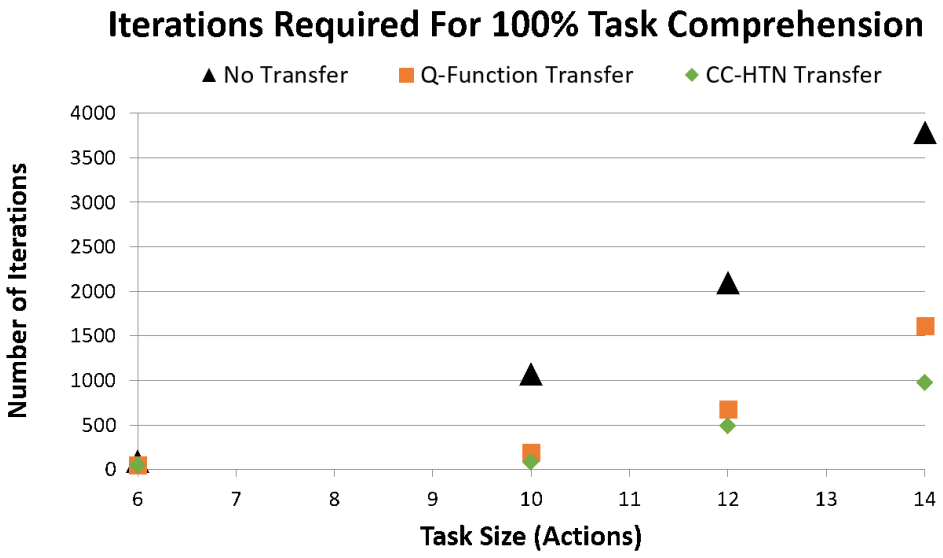

Figure 2.6: Task MDP discovery results on a dataset of 25 food preparation tasks with a primitive action set size of 39. Q-function transfer aggregates Q-functions from known task MDPs to bias action selection during exploration. CC-HTN Transfer uses Q-function transfer with macro actions learned from the CC-HTNs of other tasks in the data set.

larger steps towards a goal state, acting as a valuable heuristic that removes multiple levels of depth from the plan search each time one is utilized. Additionally, for applications where a human user is assisting the task planner by specifying action sequences manually, CC-HTN-based abstractions can greatly reduce the effort and time required to formulate a solution. As these abstractions readily indicate intra-task dependencies and can identify parallel subtasks through comparisons of required resources at each subgoal, I also obtain benefits seen in AND/OR graph constructions for assembly tasks [108].

More generally, given a task with an action set of size $|A|$ and an optimal solution length of d, a standard breadth-first search will have complexity $\mathcal{O}(|A|^d)$. With an abstraction strategy that adds *useful actions* and follows the Downward Refinement Property [115], the average search complexity will improve exponentially with the layers of abstraction provided. Given k levels of abstraction, the average complexity reduces to $\mathcal{O}([k|A|]^{d/k})$. The addition of modern heuristics (such as FFRob [120]) can be expected to provide more improvement. Thus, given the ability to determine topically similar tasks, the burdens incurred by increasing the size of the action dictionary are more than compensated for by reducing the necessary search depth. This strategy is particularly effective for robots

designed to perform tasks that originate from the same domain (such as a cooking robot or a particular type of product assembly machine), but will decrease in effectiveness as the task corpus diverges.

I conclude with results comparing three approaches to a task exploration problem to quantify my contribution's transfer learning benefits for real tasks (Figure 2.6). Using a data set of 25 food preparation tasks of between 6 and 14 steps encoded as SMDPs ($|A| = 39$), I measure the number of iterations required for an ϵ-greedy exploration policy to capture all structural insight (valid subgoal orderings) of a task. The macro actions produced by the CC-HTNs allowed the agent to achieve full task comprehension faster than either random exploration or flat Q-function transfer strategies in every trial. This indicates that autonomously built CC-HTNs are more effective than the implicit subsequence clustering of Q-function transfer, demonstrating that a deeper structural insight is captured by the generated macro actions.

The ability for an agent to learn from its past experiences to facilitate its operation in new scenarios is critically important for a collaborative robot, and rapid re-planning is essential when cooperating with agents whose behaviors are not known a priori. Intelligent strategies for selecting which abstractions to transfer across tasks are outside the scope of this contribution and remain as future work.

2.1.8 Summary

In this section I introduced CC-HTNs and provided a novel, bottom-up approach for autonomously deriving hierarchical task structure from graphical task representations, using SMDPs as a representative example. In doing so, I introduce the Conjugate Task Graph: a task representation with graphical properties that facilitate the identification of underlying structure. The approach I present is widely applicable, depending only upon the availability of primitive skill classification and ordering constraint knowledge within the target task. To make this contribution even further accessible, I identify and provide solutions to instances where ambiguity may arise in building the CC-HTN.

I evaluate this work in domains deeply relevant to human-robot teaming: collaborator goal inference via state estimation and task learning. My results suggest that CC-HTNs are

useful in segmenting complex tasks, reducing computation time and enabling inference at multiple levels of abstraction. The generated macro actions find further utility in transfer learning settings, accelerating the comprehension of new tasks and enhancing the value of prior training data. Finally, these results demonstrate that CC-HTNs can reduce the average search complexity for agents that operate in use cases where task subgoals may be shared across activities, a reasonable expectation for robots that work with humans or in human populated environments.

2.2 Discovering Task Constraints through Active Learning

Effective robot collaborators that work with humans require an understanding of the underlying constraint network of any joint task to be performed. Discovering this network allows an agent to more effectively plan around co-worker actions or unexpected changes in its environment. To maximize the practicality of collaborative robots in real-world scenarios, humans should not be assumed to have an abundance of either time, patience, or prior insight into the underlying structure of a task when relied upon to provide the training required to impart proficiency and understanding. This section introduces and experimentally validates two demonstration-based active learning strategies that a robot can utilize to accelerate context-free task comprehension. These strategies are derived from the Conjugate-Task-Graph in the previous section, a representation of a Markovian task graph that acts as a constraint network and can be used to inform query generation to guide policy exploration. I present a pilot study showcasing the effectiveness of these active learning algorithms across three representative classes of task structure. The results show an increased effectiveness of active learning when utilizing feature-based query strategies, especially in multi-instructor scenarios, achieving better task comprehension from a relatively small quantity of training demonstrations. I further validate my results by creating a set of virtual instructors from a model of the pilot study participants, and applying them to a set of 12 more complex, real world food preparation tasks with similar results.

2.2.1 Motivation

For the vast majority of situations, it is unreasonable to assume that a robot can be deployed in an environment carrying the full knowledge required to adequately perform its duties. As a motivating example, consider a household robot that is tasked with assisting a human with cooking duties. The preparation and treatment of each ingredient type likely requires its own unique trained skill (e.g., mashing potatoes vs. kneading dough) or composition of known skills. It would be impossible to precompute all possible skills required for a robot to have proficiency across all food preparation tasks, and equally difficult to develop objective functions enabling a robot to practically learn these skills on its own. Requiring a robotics expert to provide new programming for each shift in responsibilities, process, or task is often prohibitively expensive from both a temporal and monetary perspective. Further, it cannot be reasonably expected that the programmer will be a subject-matter expert for the target domain, resulting in the difficult situation of attempting to translate an expert's knowledge and learned heuristics into code.

To overcome this difficulty, Learning from Demonstration (LfD) has been developed and validated as both a popular and effective mechanism to afford non-experts the ability to easily impart new task and skill knowledge to robots [18, 121]. The ultimate goal of LfD research is to build systems capable of learning and generalizing tasks from naturally demonstrated examples by non-technical users. While many methods of skill learning can perform quite well and are considered feasible even when requiring thousands of simulated trials to converge on a proficient skill policy [9], it is clear that the applicability of LfD will be severely limited by such large training set requirements. It would be an unwise design decision to assume a human is willing or able to demonstrate dozens of executions of each possible cooking task for the sake of training a robot.

Building more capable, intuitively trainable robots is a vital step towards transitioning them from isolated, independent workers to capable, safe, efficient collaborators. To facilitate this transition, the robotics community must collectively overcome a broad variety of challenges spanning the domains of skill acquisition, implicit and explicit communication [65, 122], solo and joint task understanding, and collaborative execution [74, 123].

Figure 2.7: Collaborative Workbench platform with experimental setup, used to learn task structure from demonstrations of construction activities.

When seeking to integrate collaborative robots into complex roles with non-trivial responsibilities, developing team-oriented behaviors becomes increasingly important. To this end, researchers have developed planning algorithms that, given the constraints of a task, allow for rapid planning and ideal resource allocation [77, 78]. As robots become more integrated into human environments, it will be equally important to develop mechanisms by which a robot can leverage task structure comphrension to synchronize its execution preferences with the expectations of its teammates [5,124] and incorporate LfD-based skills into high-level planning systems such as Hierarchical Task Networks [47, 102].

Enabling these behaviors is contingent upon the agent having an understanding of the effects its actions have on the world. Skills acquired via LfD do not always have this knowledge accessible via the necessary symbolic formalization, as generating it requires considerable sensing or intention recognition abilities [125]. These effects can be dealt with indirectly, as skills may be viewed merely as functions acting on the environment, producing a new environment state given an existing one. Thus, learning which compositions of these functions are valid allows the agent to plan and act without direct symbolic representations of each component skill [126]. The problem of discovering these valid sets of compositions is equivalent to learning the constraint network for a given task.

In this section, I address the problem of a robot (Figure 2.7) learning the constraint network of a task through the observation of human instructors. Learning task structure

in this manner can require a large set of demonstrations to achieve a sufficient amount of training diversity. If an instructor behaves habitually and does not produce novel demonstrations, very little new structural information can be acquired per training session and no new constraints will be learned/removed. As an example, if an instructor were teaching a robot about salad preparation but only ever chopped carrots before chopping celery, the robot will learn an artificial constraint indicating that carrot preparation must occur prior to celery preparation in this task. Further complicating matters, instructors are expected to properly modulate the diversity of each demonstration to the learner, maintaining responsibility for maximizing learning gains at each step. To mitigate these issues, a robot can leverage its embodiment to participate in the learning process, asking the instructor questions that seek to maximize its learning gains. This process, called Active Learning, has been shown to be effective in LfD domains [33].

This section offers the following contributions:

- Two active learning query generation algorithms based on graph topology that significantly outperform a standard random exploration function in constraint learning for sequential manipulation tasks.
- A pilot study investigating how different active learning query strategies affect the learning of three representative classes of task constraint networks.
- An examination of the effects of drawing demonstrations from multiple instructors on learning task structure.
- An experiment showing that better training data may be obtained by optimizing for inspiring diverse instruction from a small group of instructors rather than extensive training from a single instructor.

2.2.2 Approach

Applying the Conjugate Task Graph transform (Algorithm 1) affords advantages to a robot learner by exposing graph features that may facilitate task structure comprehension and discovery. I show that the CTG can be used within the context of active learning to achieve

accelerated understanding of a complex task structure given a realistic and limited set of demonstrations and instructors. I focus particularly on learning a constraint network for each task, indicating acceptable skill sequences to reach goal states for the activity. In doing so I remove the need for complex intention recognition or symbolic formalization of skills, making this approach particularly accessible to LfD-based systems.

I focus especially on the domain of tasks and environments where instructor availability is limited (low demonstration count), human-robot communication is heavily constrained (no explicit communication from human to robot), and collaboration with other trusted robot instructors is limited or unavailable (low collaborator count). I maintain these very conservative restrictions in an effort to remain relevant to a maximally diverse range of collaborative robots.

2.2.2.1 Platform

This research was performed using a robotic workbench designed for human-robot collaboration (Figure 2.7). The Collaborative Workbench is a 1.83m x 0.762m worktable equipped with two KUKA YouBot 5-DoF light manufacturing arms spaced 0.5m apart, with approximately 0.5m of overlapping operational envelope. Each arm is equipped with a parallel bar gripper, suitable for "pick and place" type tasks. A remote system interface is provided via tablet computer.

2.2.3 Learning Environment

This work focuses on using active learning and LfD to discern the underlying structure of a task without context. I represent a task SMDP using standard graphical notation $G = \{V, E\}$. I define an observed task execution x as a simple directed graph with membership in the set X consisting of all valid task execution graphs. Thus, $x = \{V_x, E_x\} \in X$ describes a single path through a set of vertices in V representing environment states linked by edges labeled with known, executed actions, terminating in a vertex describing a goal state (terminal vertex). I define a task as the weakly connected directed multigraph $T = \{V, \bigcup_{i=1}^{|X|} edges(x_i) \in X\}$, describing a Semi-Markov Decision Process. A vertex in this graph is labeled with information representing the state of the environment, while directed

edges connecting environment states are labeled with actions that must be performed to transition between them. This typical SMDP representation is subsequently referred to as an *Environment-space Graph* (Figure 2.8—top).

Formally, the task structure discovery problem can be formulated as one of graph exploration. The learner is presented with an ever-growing library of observations L of successful skill execution sequences ($l \in L \implies l \in X$), successively gaining more insight into the task being performed as novel elements are added. Certain paths through the graph are more valuable than others, as a demonstration primarily serves to reveal new vertices and edges. As such, a learner is motivated to encourage an instructor to provide the most informative path per demonstration, desiring the instructor to show a demonstration x according to

$$\arg\max_{x \in X} |unique\_edges(L \cup x)|$$

2.2.3.1 Multi-Instructor Task Learning Domain

As robots become increasingly ubiquitous, the availability and magnitude of potential gains from distributed task learning across multiple instructors increases.

It is likely there will be a substantial overlap of training needs across collaborative robots. Across every robot trained in an instance of the general cooking task, some subsets of them will be trained for the same particular instantiations of the activity. Leveraging the data from these instructors with similar robots, environments, and task demands can be mutually beneficial, but also potentially dangerous if the shared data is invalid or maliciously constructed. For this reason, even as robots become commonplace in many task domains and the amount of relevant collaborators increases, it may not be a safe assumption that all others' training data should be implicitly trusted. This reinforces the need to expect situations in which there may be limited numbers of available collaborators. I define the multi-instructor task learning domain as m independent instances of the *task learning domain* whose observation libraries $L_1, L_2, ..., L_m$ can be joined and re-distributed to each participating instructor, leveraging not only additional demonstrations, but benefitting from the diversity inherent to multiple teaching styles. For the experiments in this section, this step occurs subsequent to all training activity to avoid the assumption that

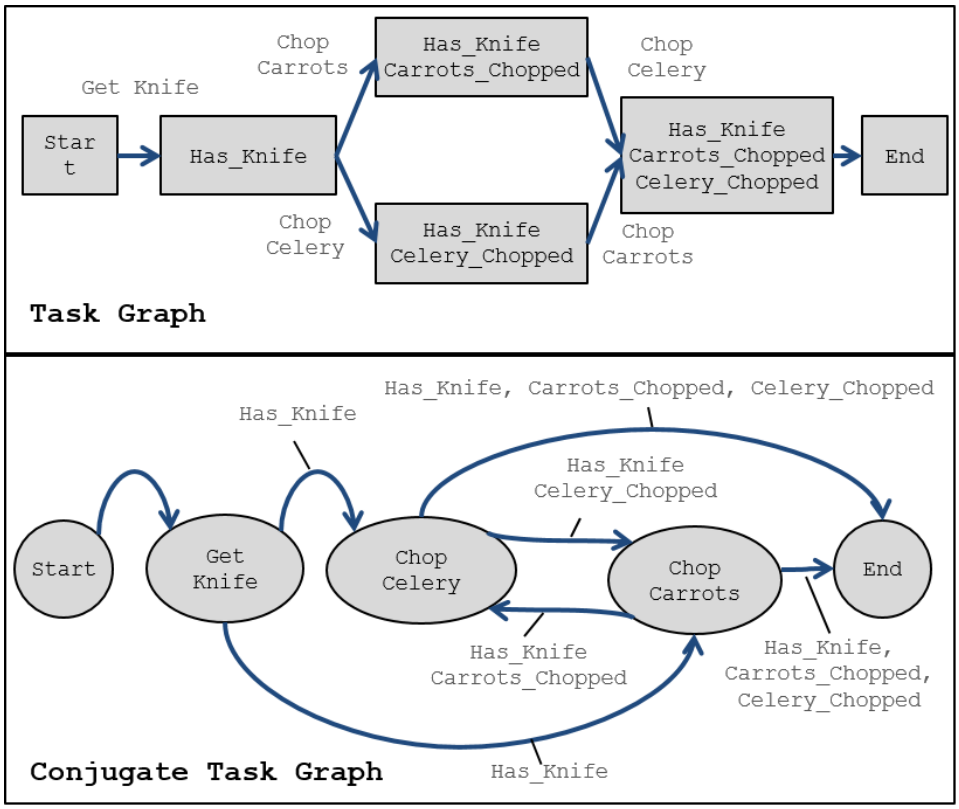

Figure 2.8: Visualization of graph transforms on an everyday task. For this task, the knife must be acquired prior to chopping, though the order that the carrots or celery is chopped does not matter.

collaborator data was available during the initial training of a task.

2.2.4 Activity Description

The work in this section focuses on learning the underlying structure of a collection of representative construction tasks. These tasks are defined through ordering and prerequisite conditions on a set of eight uniquely colored interlocking blocks (Figure 2.7). Block construction tasks were chosen as they can be flexibly or rigidly defined and will not bias participants towards particular demonstrations in a way that tasks involving familiar objects might. Participants were only permitted to construct from the bottom up in their demonstrations, resulting in a total possible task space of 40320 block usage sequences. The three representative tasks chosen to test the active learner query algorithms are drawn from three distinct structural compositions (Figure 2.9).

2.2.4.1 Workspace

The workspace for performing tasks on the Collaborative Workbench contained one KUKA YouBot arm, eight uniquely colored building blocks, a piece of paper with task descriptions, and a tablet computer (Figure 2.7). The leftmost YouBot arm was not utilized for this experiment. Each of the eight blocks had a labeled place-card designating its home position affixed to the table within range of the arm (the "resource area"). The tablet, placed next to the resource area, presented an interface with buttons corresponding to each of the blocks on the table. The area immediately in front of the participant was informally designated as the construction area. The paper with task descriptions and reference images was placed just left of the construction area.

2.2.5 Learning System

A typical passive learner would gain knowledge about the structure of a task by observing performances of it. Duplicate observations would not provide new insight into the underlying structure, though they may be helpful for determining user preferences or reasoning about commonly occurring orderings. This method, while non-intrusive, can be inefficient if the provided training data is not suitably diverse. One practical method of overcoming this challenge is to participate in the learning process.

2.2.5.1 Active Learning

In the general case, Active Learning involves a learner generating or selecting examples to be labeled by an oracle. With a means to participate in the instruction process, a learner is capable of increasing the effectiveness of the tutoring process if she can guide an instructor to provide and label useful examples specific to her current internal state [127, 128].

Within the task learning domain, useful examples are those which inform the learner about previously unknown constraints between the components of a task. When learning task structure, active learning provides a mechanism by which a learner can suggest a potential edge from the environment-space graph via a demonstration query, implicitly specifying the originating vertex (current environment state) and explicitly providing a

skill label (the query itself). For example, a robot may ask a human chef if the flour can be added after adding butter in a cookie recipe. In doing so, the learner is requesting the oracle to reveal an edge leading from a vertex containing the current state of the food to a nearly identical vertex where the flour has been added. An instructor can respond to this query positively by utilizing the suggested skill and adding flour, or negatively by performing a different skill such as adding sugar instead.

While a negative response does not necessarily indicate that no edge with the queried label exists from the current environment state, it does suggest that it is unlikely. A particular edge that is known or confidently assumed to not exist can be labeled as an *anti-edge* in the environment-space graph, informing the learner that he should assume no transition is possible matching that particular environment state/skill combination. Anti-edges serve an important role within the task learning domain, as queries are a limited resource [129] and should not typically be used to test previously ruled-out transitions.

As individual instructor habits and preferences can diminish task demonstration diversity, active learning provides a mechanism by which a robot can safely participate in the learning process, encouraging the presentation of varied examples. I show that the process of solving the task learning problem can be expedited with the assistance of an intelligent strategy for encouraging demonstration diversity through active learning, derived from the properties of the *Conjugate Task Graph* (CTG).

While the environment-space graph provides an easily followed execution plan for an arbitrarily complex task, it does not immediately convey features helpful for either discerning task structure or informing an active learning strategy to inspire useful demonstrations. To gain easily interpreted features for satisfying these goals, I utilize the CTG. In contrast to the environment-centric view of the standard graphical task representation, the CTG provides a skill-centric overview of a task.

As described in the previous section, given a standard environment-centric task graph $G = \{V, E\}$, the Conjugate Task Graph is a directed multigraph $H = \{W, F\}$ consisting of a unique set of vertices $W = \{\text{label}(e) \mid \forall e \in E,\ \text{label}(e) \notin W\}$ and a set of edges F connecting vertices in W with labels corresponding to prerequisite environment features derived from the postconditions of previously executed skills (Figure 2.8).

The CTG provides several benefits when compared against the environment-space graph in the context of selecting a skill for a demonstration query. Simple features derived from a partially known CTG such as skill distance and skill connectivity can be used to beneficially inform an active learning query mechanism.

2.2.6 Query Strategies

For realistic training scenarios, it is important for an active learner to utilize query opportunities as a scarce and valuable resource, optimizing the value derived from each chance to interact with the instructor. Posing too many queries can slow down the training process and potentially cause the instructor to reduce the number of demonstrations or even abandon the training entirely.

As the goal in the task learning domain is to achieve maximum diversity of training examples, the value of a particular query becomes less obvious than merely evaluating whether or not a particular skill can be performed immediately given the current environment state. In some cases, it may prove more valuable to inquire about a skill that cannot be utilized immediately from the current state, with the consequence that the instructor may demonstrate a unique path from the current state to the queried skill. I evaluate the effectiveness of three different active learner query strategies: random, distance-based, and connectivity-based.

2.2.6.1 Random Querying

The baseline mechanism that I compare my strategies against is random selection. The random query mechanism selects a subgoal at random from the set of subgoals that have not yet been met during the current demonstration of the task. This set excludes the set of skills and subgoals for which direct transitions are known from the current state. Additionally, the random query mechanism made use of anti-edges, avoiding repetitive or uninformative queries.

2.2.6.2 Distance-based Querying

The distance-based query obtains a distance score for each vertex in the CTG (transformed from the learner's current environment-space graph), measuring the shortest path to each. This query mechanism selects the closest skill that has neither been executed nor describes an existing transition from the current state in the environment-space graph. Transitions that are eliminated by anti-edges are also ignored. Ties are broken randomly.

2.2.6.3 Connectivity-based Querying

The connectivity-based query utilizes the degree of each vertex in the CTG and its edges' constraint properties to rank potential queries. Each inbound edge for a particular vertex is scored inversely proportionally to the number of environmental prerequisites on its label. For the experiment, I set a base score for each edge equal to the total number of possible prerequisites (obtainable by examining the inbound edge constraints for the terminating vertex of the CTG). Inbound edges are given scores of [base_score − prerequisite_count], while outbound edges on a vertex are scored at a flat value of half the base score. The score for a vertex is defined as the sum of its inbound and outbound edge scores. These values were chosen to prioritize well connected vertices with minimal prerequisites. As in the distance-based and random queries, skills that had already been executed this iteration are not considered for selection. Anti-edges are also utilized to avoid known negative queries. The resulting demonstration query is chosen as the remaining skill with the highest score. As in the distance-based query, ties are broken by a skill being chosen randomly from the set of best scoring results.

2.2.7 Experiment Design

To explore the performance impact that each of the three query strategies has on a robot utilizing active learning to learn the structure of a task, I conducted a pilot study. Participants were instructed to perform construction tasks in front of the robot, with the expectation that the robot will occasionally ask questions about the current activity. The experiment involved each participant performing 42 demonstrations consisting of 14 exam-

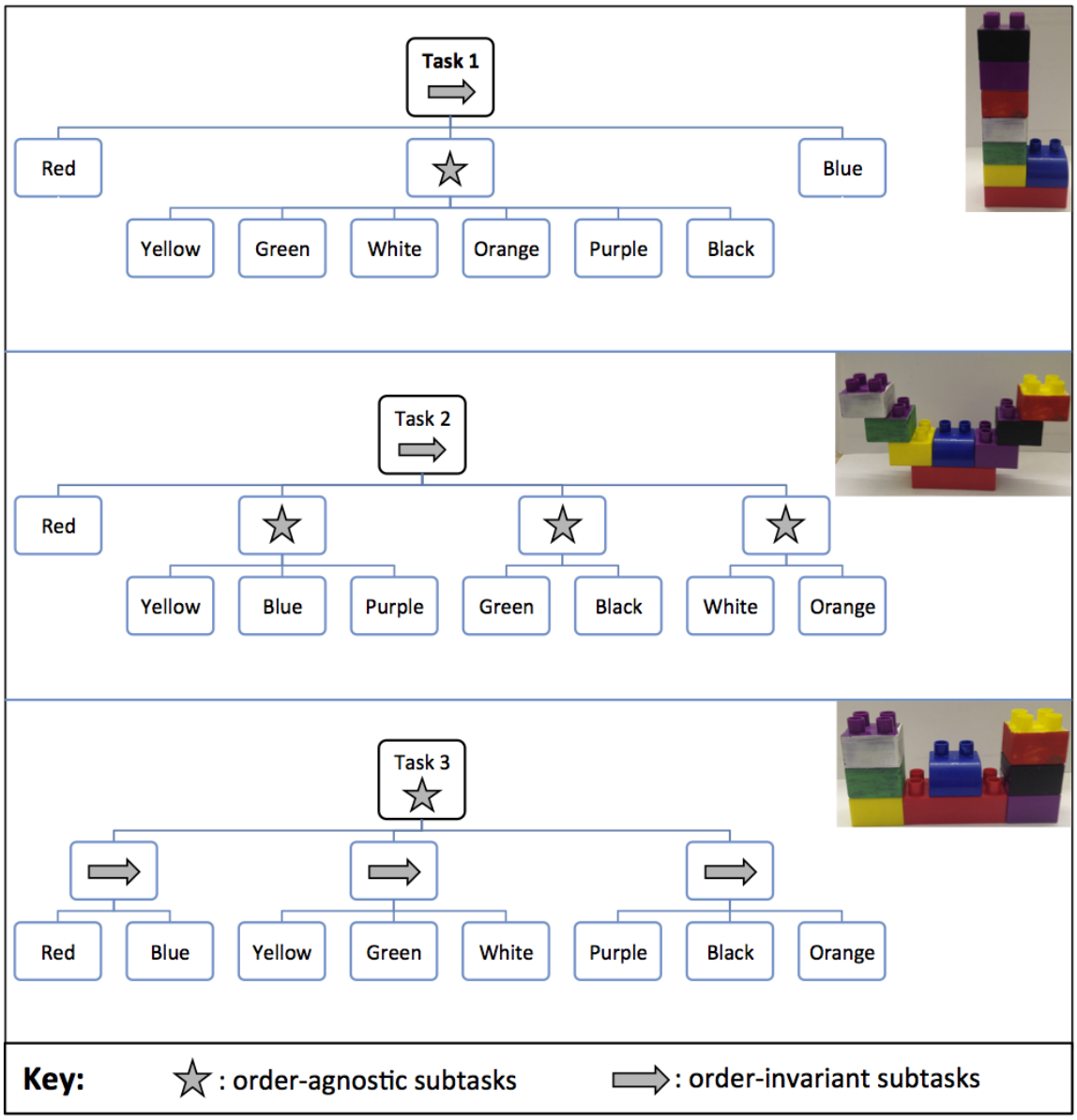

Figure 2.9: Construction activity task hierarchies

ples for each of three different tasks. There were four algorithmic conditions, including one passive learning condition that did not query and three active learning conditions: "Random query", "Distance-based query", and "Connectivity-based query". In querying conditions, the robot only posed open-ended, material-centric demonstration queries. This took the form of a request to use a particular material (building block) next.

For each active learning algorithm, the participant performed each task in succession (task 1, then 2, then 3). Tasks were demonstrated in an interleaved sequence to provide

a more naturalistic setting in which to evaluate task instruction, as I seek to investigate a casually instructed robot rather than one with a dedicated instructor expending a concerted effort teaching it. Participants only completed two performances of the task sequence for the "No query" condition, as it was included to familiarize the participant with the experimental procedure. Each of the three active learning conditions involved the participant performing each task four times.

2.2.7.1 Tasks

The first task incorporates a large group of order-agnostic elements within a rigid ordering sequence, identifiable as a chain containing a large clique. Participants were instructed to construct a tower structure (Figure 2.9-top) with the constraints that the first block placed must be red, the last block placed must be blue, and the order of the other blocks {Yellow, Green, White, Red, Purple, Black} did not matter. This task has 720 valid execution paths. Queries were made after the first, third, and fifth steps of this task.

The second task is an order-invariant sequence of small order-agnostic subtasks, a chain of cliques. Participants were instructed to build a V-like structure (Figure 2.9-middle) from the bottom up, one row at a time. The rows were (in increasing order): {Red}, {Yellow, Blue, Purple}, {Green, Black}, {White, Orange}. This task has 24 valid execution paths. Queries were made after the first, third, and fifth steps of this task.

The third task consists of an order-agnostic trio of order-invariant subtasks, otherwise described as a clique of chains. Participants were instructed to build three different towers {Yellow, Green, White}, {Red, Blue}, {Purple, Black, Orange} in any order, with the restriction that once a tower is started it must be completed before beginning another (Figure 2.9-bottom). This task could be completed via six unique execution paths. Queries were made at the beginning of this task, as well as after the second and fourth steps.

2.2.7.2 Procedure

For the duration of the experiment, the participant was seated at the workbench in front of the robot. After being briefed that he would be demonstrating a set of tasks for the robot, the task descriptions were explained to him with the opportunity to ask questions

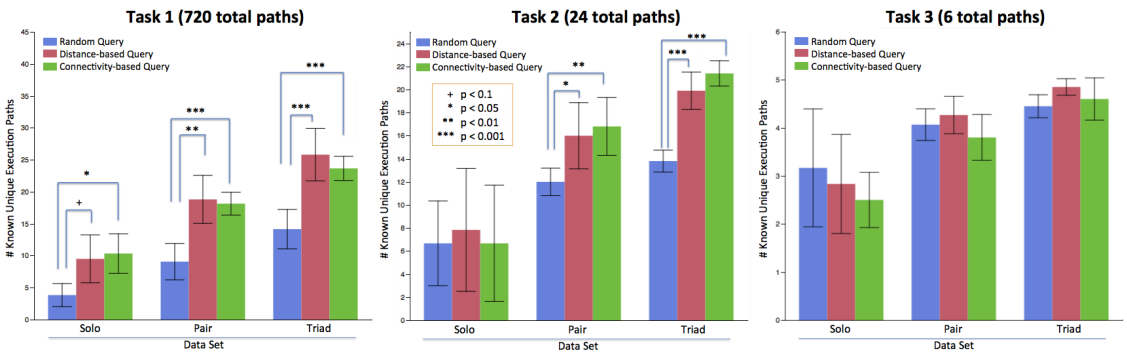

Figure 2.10: Active learning strategies compared across tasks and data sets.

about the tasks to the experimenter. Participants had access throughout the experiment to a description for each task that included a reference image.

Participants were motivated to complete each series of tasks quickly in an effort to encourage natural demonstrations rather than carefully considered training examples. The tablet interface on the table informed participants as to which task they would be performing, as well as when a particular demonstration ended and their structure could be disassembled. Participants were informed that the robot would be testing four different learning algorithms, and that the robot may occasionally ask questions of them.

During a task performance, participants would use a block in their demonstration then press the corresponding button on the tablet to inform the system of the utilized block. Upon making a selection on the tablet, the screen would fade to gray if a query were about to be performed, so the participant would know not to continue until the robot finished its action.

Subsequent to the first demonstration for a each algorithm, the robot would make queries. To perform a query, the robot would move from its resting position and orient itself such that it would be pointing at the block required for the queried skill with its parallel bar gripper. Once the pointing behavior was completed, a synthesized voice asked "Can you use this one next?" before the robot returned to its resting pose. After all robot motion was complete, the tablet screen would display the normal interface and allow the participant to continue. Participants were never given instructions regarding the handling of the robot's inquiries.

Participants were not told any details about the algorithms other than that four were

being tested. When the algorithmic condition changed, participants were informed via the tablet interface that the robot would be forgetting everything it had just learned so it could accurately compare methods later.

2.2.8 Results

Six participants (4 female, 2 male) completed the experiment, each providing 42 task demonstrations. Each participant completed the experiment in about 55 minutes. The data set consisting of the learned task graphs trained by individuals is referenced as the *solo* data. A second data set, referenced as the *pair* data, was created from combining task graphs for each possible unique pairing of participants ($\binom{6}{2} = 15$ pairs). A third data set, labeled *triad* data, was created through the combination of every possible unique outcome from selecting three participants ($\binom{6}{3} = 20$ triads). I evaluated my data with the purpose of examining effects that the different active learning strategies have for the three task classes chosen, in the cases of either a single instructor training in isolation or a small group of independently collected, pooled instructor data.

I use the number of known unique paths through the environment-space graph from the origin vertex to the terminal vertex as the primary metric for determining the degree of task comprehension achieved with each query mechanism. Statistical results across were obtained using one-way ANOVAs on path count (dependent variable) and algorithmic condition (independent variable), separated by task and number of instructors (figure 2.10).

2.2.9 Individual Training

For the single instructor demonstration data (N=6, total demonstrations per task=4), only task 1 yielded significant differences between query strategies. A one-way ANOVA with number of known unique task execution solutions (paths through the SMDP) as the dependent variable and query strategy as the independent variable for each task (1,2,3) and instructor condition (single, pair, triad) was run on the data. For the single instructor case of the first task, my analysis shows a significant effect of query strategy ($F(2,10) = 11.508$, $p < 0.01$). Post-hoc tests with a Bonferroni adjustment indicate a significant difference between the random and connectivity-based algorithms (6.5, $p < 0.05$), as well as a

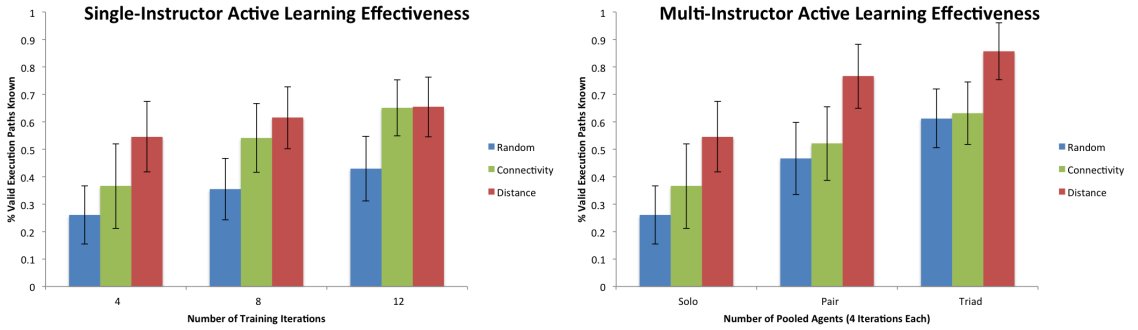

Figure 2.11: Results for simulated active learning strategies, averaged over a set of 12 kinesthetically trained food preparation tasks.

marginally significant difference between the random and distance-based algorithms (5.667, $p = 0.055$). No significant difference was found between algorithms for tasks 2 and 3 in the single instructor scenario.

2.2.10 Joint Training

I investigated the effects of combining instructors' data to measure the utility of the feature-based query mechanisms on multi-instructor scenarios. Such scenarios are particularly relevant to robots that learn from demonstration, as I seek to characterize the demonstration diversity that can be obtained from pooling instructors together with respect to the active learning strategy used. The pilot study results show that the feature-based query strategies encourage diversity across teaching styles, leveraging each instructor's a priori task execution preferences to prompt novel demonstrations. I show that this diversity gained from pooling instructors can be more beneficial for learning task constraints than using a single instructor with the same amount of demonstrations.

In this collaborative training scenario, feature-based query strategies perform substantially better than the random query condition. Most importantly, the distance-based query learner completely learned the structure of the second task and the connectivity-based strategy learned the second and third task entirely. This result is motivating, as it suggests that particular query strategies can greatly amplify the benefits of joining different teaching styles. I proceeded to analyze this impact by creating and comparing the pair and triad data set for each task and querying algorithm.

In the pairs data (N=15, total demonstrations per task=8), there is a significant effect of algorithm type for task 1 ($F(2,28) = 17.381$, $p < 0.001$) and task 2 ($F(2,28) = 9.591$, $p = 0.001$). Post-hoc tests on the results for task 1 indicate significant differences between random and connectivity-based algorithms (9.1, $p < 0.001$) and between random and distance-based algorithms (9.7, $p < 0.01$). Post-hoc tests on the results for task 2 indicate a significant difference between both the random and connectivity-based algorithms (4.8, $p < 0.01$) and between the random and distance-based algorithms (4.0, $p < 0.05$). In the second task, the joint instruction process produced a training graph that has a full understanding of the task structure after only 8 demonstrations spread across two non-interacting instructors (4 demonstrations each). The overall performance increases in the third task, but it is unclear that the improvement had any connection to the querying process rather than merely the higher demonstration count, likely due to the low number of unique paths in the graph (6 total).

Algorithm choice had a significant effect in the instructor triad data (N=20, total demonstrations=12) for task 1 ($F(2,38) = 25.107$, $p < 0.001$) and task 2 ($F(2,38) = 85.597$, $p < 0.001$). In the first task, significant differences were again found between random and the other two algorithms: (9.5, $p < 0.001$) for connectivity-based and (11.65, $p < 0.001$) for distance-based. In the second task, there were significant differences found between the random exploration method and the CTG based algorithms: (7.6, $p < 0.001$) for connectivity-based and (6.1, $p < 0.001$) for distance-based.

2.2.10.1 Further Analysis

The results of the first experiment suggested that dividing task demonstrations across multiple instructors may be ideal for maximizing training diversity. In this section, I not only seek to validate my previous results with real-world tasks, but also seek to investigate more directly comparable scenarios where training demonstration counts are kept constant but the number of instructors are varied across query algorithms. In this second experiment, I evaluated the performance of my query generation algorithms on learning a set of 12 real-world inspired tasks. These tasks, from the food preparation domain, contained more steps and more structural complexity than the construction activities from the pilot study,

containing 10 to 14 skills and 12 to 24 unique execution paths each.

To perform this analysis, I modeled the pilot study participants' behaviors regarding preferred demonstration sequence, tendency towards unprompted demonstration deviations, and responses to robot queries for each task type and built a simulation environment. An analysis across participant data showed queries that could be accommodated within 3 steps were heeded by the instructor, adjusting her demonstration to cover the queried skill at the earliest allowable time. Queries concerning skills beyond that time horizon were disregarded and had no demonstrated effect on instruction. Participants were likely to remain true to their initial demonstration sequence for each task across algorithms, though their choice of initial sequence appeared to be motivated by the distance of blocks from them on the table (subject to small seemingly random perturbations for the blocks furthest from the user). In the simulation, I assign each instructor a preferred demonstration path with small random perturbation based upon a fixed arbitrarily chosen canonical base path. Instructor data was pooled in the same manner as the pilot experiment: after all trials were completed, the union of the resulting learner graphs was evaluated.

I simulated 50 trials of learning each task with conditions identical to the pilot study (4 demonstrations per task, 3 queries allowed per demonstration, no queries during the initial demonstration). The simulation involved the production of valid action orderings by virtual instructors, influenced by queries posed from the simulated robot agent. The robot's learned task graph was constructed with a procedure identical to that used in the live experiment. I compare results between querying strategies in the multi-instructor and solo instructor scenarios, presenting the mean percentage of task comprehension in figure 2.11. In the single instructor scenario, I find average task comprehension rates of 25.9%, 35.4%, and 42.9% for the random query strategy at iterations 4, 8, and 12, respectively. The connectivity-based query generator posted average performances of 36.5%, 54.1%, and 65%, while the distance-based learner achieved comprehension rates of 54.6%, 61.5%, and 65.4%. For the multi-instructor scenario, the random learners received demonstrations for 25.9%, 46.6%, and 61.2% of the possible solutions in the graph. The connectivity-based learners were taught 36.6%, 52%, and 63.1% of graph solutions. Most effective in this experiment, the distance-based algorithm produced average comprehension rates of 54.6%, 76.6%, and

85.8% across all tasks.

These results further highlight the result that these querying strategies are not only effective, but effective in different ways for each instructor. This is evidenced by the increased task network coverage achieved by pooling groups of instructors together. Had the query strategies converged instructors to the same teaching strategy, the pooled data would not show an increase in diversity when compared against the single instructor scenario (with identical total demonstration count). These results also support the justification of using a multi-instructor setup over a single instructor, even when provided the same total training time, as the teaching style of each instructor is expanded in non-convergent ways to cover more of the task graph.

2.2.11 Discussion and Conclusion

The presented results show that feature-based query strategies derived from Conjugate Task Graphs are a promising avenue to explore within the task learning domain, especially under the restriction of limited data collection. Task structure plays a large role in determining the potential effectiveness of a query, but did not differentiate between the two tested methods enough to suggest a clear winner in the pilot study. In my simulated validation on tasks derived from real world activities, the distance-based querying mechanism showed clear superiority in the multi-instructor domain.

The CTG provides a set of simple features that can be used to inform an active learner's query generation, obtaining more statistically effective training data from a human instructor than a traditional random exploration strategy. I performed an analysis on the effectiveness of combined instruction from the pilot study, looking at the effects of creating pooled data from small teams of instructors. This result was validated via a simulation of instructors based on a model inspired by the pilot study participants, using a set of 12 complex tasks derived from real-world activities.

The simulation results reinforce the notion that, when seeking to learn the constraint network of a task, utilizing multiple instructors provides an implicit demonstration diversity benefit over the single instructor case, even given the same total demonstration quantity. This diversity of user preference enables a robot to more quickly learn valid skill sequences

by working with each instructors' unique teaching style to produce a wider range of demonstrations. By increasing the effectiveness of training demonstrations, I provide a way to reduce the human-oriented expense of providing a robot with training data for a given task. The query strategies utilized encouraged instructors to diversify their examples significantly more, the benefits scaling well with the number of instructors.

These results suggest that learning complex task network constraints from limited quantities of demonstrations is feasible through small-scale collaboration when an effective active learning query strategy is used. The described contributions are particularly applicable to collaborative robots that learn from demonstration, helping to remove some of the barriers to achieving task proficiency that have rendered more autonomous learning techniques infeasible.

Chapter 3

Supportive Behaviors

In this chapter, I focus on algorithms that enable a robot collaborator to support its teammates' task execution. Central to this assistance are *supportive behaviors* which I define as optional, off-goal actions that contribute indirectly towards more rapidly satisfiable and/or less difficult task solutions.

Particularly within assembly domains, there exist actions that can be performed to ease cognitive or physical burdens that don't directly contribute towards the completed construction. One such example is organizing circuit components into discrete piles, such that all wires are axis-aligned and together, or physically sorting resistors in ascending order of resistance. By performing these organizational behaviors that are tangential to the task goal, future part-seeking actions are facilitated and the overall makespan for the assembly task may be reduced. For motion planning, actions that remove obstacles from the environment of an object grasping behavior will reduce the problem complexity by allowing for more valid or possibly less complex motion paths.

Similarly, positioning objects near their final positions (e.g., screws near appropriate guide holes, components in an approximation of their final configuration, etc.) may reduce the cognitive load on the lead agent actually performing the assembly, as it becomes less ambiguous how to assemble the components. Within the TAMP domain, this is equivalent to reducing the symbolic planning complexity. The key insight being leveraged involves organizing the environment into a configuration favorable to a lead agent's planning heuristics, such that the agent's search order over objects to use for action parameterization is

optimized.

3.1 A Taxonomy of Supportive Behaviors

I classify supportive behaviors as generally belonging to one of five high-level categories, enumerated below. While this is by no means an exhaustive list of supportive action types, the close proximity, shared workspace environment I focus on makes these specific classifications particularly relevant. The remainder of this section focuses on policy learning and transfer for robot actions in the areas of: materials stabilization, materials retrieval, collaborative object manipulation, enhancing awareness, and task progression guidance.

3.1.0.1 Materials Stabilization

This behavior class involves having the robot hold an object maintaining a particular pose. The inherent goal of such an action is to position and orient an object such that the primary worker can more easily perform the target task. Examples of this include holding fabrics together to allow a human to sew them more easily, orienting a wire and PCB in such a way that they may be soldered together, or maintaining the position of a piece of furniture so a component may be pressed into place.

3.1.0.2 Materials Retrieval

The retrieval class of behaviors involves changing the available workspace resources such that a worker needs to exert less effort or cognitive load to find necessary parts or tools. Actions in this class can range from non-intrusive methods such as pointing, to minimally intrusive actions such as placing objects conveniently nearby, to directly interactive handover actions.

3.1.0.3 Collaborative Object Manipulation

For particularly unwieldy materials or in instances where precision is especially important, a robot assistant may render help in the form of moving objects in tandem with another worker. Two particularly salient use cases for these behaviors are the placement and orientation of heavy or large objects and the guidance of tools to effect other materials. In

the case of the latter, one can imagine situations that may arise in complex tasks where motion constraints on particular tools will greatly reduce the difficulty of a subtask. A concrete example of this may be an assistive robot enforcing a planar constraint while a human manipulates a jointly held sanding tool or enforcing a depth constraint when etching a fragile surface with a dremel tool.

3.1.0.4 Enhancing Awareness

There are behaviors that do not directly manipulate the environment, yet may also be of exceptional assistance to a worker. One example of an awareness-enhancing behavior would be a robot manipulating a camera to provide an alternative, more helpful perspective to a worker. Other, less complicated actions in this category may include using a flashlight to illuminate areas inside a part being worked on, or directing a laser pointer to indicate relevant through-hole paths on a PCB that is being wired.

3.1.0.5 Task Progression Guidance

An assistive robot may also perform actions that inherently guide the progression of a task towards a particular action sequence. Unlike the other four classifications of assistive behavior, influencing the ordering of subtask completion involves reducing the cost (in terms of time or effort) of a collaborator pursuing certain actions over others. One minimal example where this behavior can be applied is a drilling task, where it may be more efficient for a worker to place all of the screws in pilot holes before drilling them all, rather than placing and drilling them one at a time. For example, a supportive robot can manipulate the course of action by performing materials retrieval actions giving the worker screws rather than the drill, or even by moving the drill further from the active work area until the screws are placed. The error mitigation weighting scheme discussed in the following section particularly optimizes for this type of support.

3.2 A Task and Motion Planning Approach

In this section, I present an algorithm that enables a robot to improve the performance of its collaborators during the execution of sequential manipulation tasks. The proposed agent-decoupled, optimization-based, task and motion planning approach merges considerations derived from both symbolic and geometric planning domains. This results in the generation of supportive behaviors that reduce cognitive and kinematic burdens experienced by collaborators during task completion. I describe the algorithm alongside representative use cases, with an evaluation based on solving complex circuit building problems. The section concludes with a discussion of applications and extensions to human-robot teaming scenarios.

3.2.1 Introduction

Real-world manipulation tasks typically involve handling and reconfiguring multiple objects to accomplish abstract goal criteria. These tasks involve a high-dimensional action space, making satisfying solutions potentially very difficult to compute. The use of traditional symbolic planning to derive a task solution may produce sub-optimal or infeasible physical impositions on an agent, rendering it insufficient for many tasks of interest. Avoiding this problematic detachment from the physical world by using a motion planner to search and evaluate feasible actions within the task space yields a separate set of issues, as this approach can make searching the task space very computationally expensive.

Task and motion planning (TAMP) addresses these challenging problems, merging the task-level goal demands with the physical reality and geometric complexities of motion planning [130]. Even with the state-of-the-art in TAMP approaches, many tasks remain intractable within online planning scenarios for a variety of reasons, including object density, goal flexibility, and kinematic complexity. These issues become even more prominent when considering multi-agent problems, as geometric back-tracking can be prohibitively expensive [131]. Multi-agent approaches to TAMP generally assume an 'equal-partners' coordination paradigm, where agents are jointly working towards directly achieving a shared goal, such that the same planning heuristics can be used across the team.

In this work, I propose a novel approach for use within multi-agent TAMP domains, utilizing a 'leader-assistant' teamwork paradigm, explicitly enabling supportive roles for agents to assume. These supportive roles facilitate task completion for 'leader' agents (that employ traditional TAMP solution mechanisms) by performing actions to reduce the cognitive (symbolic planning) or kinematic (motion planning) difficulty of the on-task actions they perform. Such roles are critical for human-robot collaboration [123].

In particular, I focus on a class of activities known as *sequential manipulation tasks*, where high level goals are achieved through a sequential series of object manipulations. This broad class of tasks is relevant to personal assistant robots and industrial robots alike (despite vast differences in expectations, capabilities, and tooling), making it an excellent candidate for application to the leader-follower teaming paradigm. In particular, the representative sequential manipulation task utilized in this paper is that of circuit building within a spatially constrained environment. Given a collection of resources (circuit pieces such as wires, resistors, LEDs, etc.), there exists a set of motor action sequences that each yield a constructed circuit with the characteristics specified by the task's goal state.

In addition to these motor behaviors that result in the successful assembly of the circuit, there exist actions that facilitate task completion without directly contributing towards reaching the goal state. These supportive actions (such as removing an obstacle on the table) contribute indirectly towards a more rapid or less challenging task solution. These behaviors are not nearly as well studied as their on-task counterparts, but become increasingly relevant as human-robot teaming and teams of robots with heterogeneous capabilities become more commonplace.

In the remainder of this section I provide a brief survey of related work, formally introduce the selected evaluation domain, and introduce a novel algorithm for improving team performance in sequential manipulation tasks within the leader-assistant teamwork paradigm. This algorithm accomplishes its purpose through the evaluation, selection, and execution of actions with the intent to optimize a task's environment for a collection of potential execution policies. By optimizing towards the maximization of a collaborator's performance, it is possible not only to improve makespan time and to reduce a partner's cognitive load, but also to manipulate them into making better decisions towards following

more optimal policies. This algorithm allows a supportive robot to drive its team to perform well even in the absence of intra-agent communication, with only coarse approximations of collaborators' kinematic abilities, while only assuming rational, goal-aligned teammates. Finally, the section concludes with an evaluation within a circuit building domain, exploring algorithm performance and behavior.

3.2.2 Related Work

Sequential manipulation tasks and multi-agent teamwork are rich research topics that are of great interest within the robotics community. Solving task and motion planning problems is well known to be a challenging area, with a great deal of tools and approaches developed to leverage more powerful abstractions and heuristics for the purpose of reducing search complexity [4, 132] or broadening the types of constraints that can be handled [133–136].

The majority of multi-agent work within TAMP problems has focused on an 'equal partners' paradigm of teamwork, producing innovative solutions that rely on the decomposition of a task into subcomponents that can be distributed across agents to maximize efficiency and capability [119, 137–139]. The ability to convey intent and to learn from or anticipate the behaviors of others is also critical to multi-agent TAMP. Accordingly, recent work on collaboration has been published with a focus on motion planning that implicitly conveys intent [73], as well as motion planning that accommodates the expected motions of others within an interaction [67].

Solving these TAMP problems is also of particular interest to the scheduling research community. Recent work in multi-agent scheduling has yielded a constraint-based method of performing extremely fast scheduling [12] that can be applied to heterogeneous teams of agents once a TAMP solution is translated into a simple temporal problem [36]. Each of these presented works handles an important subset of the issues central to multi-agent task and motion planning, including the management of diverse capabilities, world models, agent models, and behavioral expectations. My contribution builds upon these considerations by broadening the utility of agents, even those without the capability to directly achieve task goals, by developing a formalized means of generating supportive actions to assist those that can.

3.2.3 Task and Motion Planning Domain

The problem domain is described via traditional planning operators and symbolic predicates (similar to [140]) with the addition of special functions to handle geometric constraints. Goal states are described by these abstract predicates, providing a description of a desired world state that may include concepts that do not clearly map to a physical description of the individual components that contribute to it (e.g., a predicate *explicitly* requiring a circuit to be closed that happens to contain a diode *implicitly* requires the diode's anode and cathode to be properly connected). Operators are representative of robot motor primitives, parameterized by potentially high-dimensional continuous values that may indicate kinematic positions, grasp types, or object poses. I address the parameterization issue through the standard practice of discretizing the parameter space via sampling.

Despite discretizing these variables into finite domains, it remains impractical to enumerate the symbolic effects of each possible action, as second-order, third-order, or arbitrarily distant effects may impact aspects of the environment (such as object reachability or placement eligibility). As such, I implement a similar strategy in keeping a state representation with predicates comprised of static literals alongside geometric state-dependent predicates that are computed on-demand.

Recent work has introduced planning concepts that greatly speed the execution-time computation of valid plans by leveraging clever insight into the underlying geometric requirements for TAMP problems [132]. One of these insights is the need to efficiently characterize the free configuration space of the robot and the manipulable objects within the scene, which is not possible given static preconditions. Accordingly, in addition to classical static planning literals, I incorporate geometric conditionals to ground the symbolic planner within the physical world as it seeks task solutions.

Choosing optimal behaviors within this problem domain is challenging to accomplish in acceptable timeframes and can easily be intractable. Generating plans via symbolic planning can typically be accomplished within reasonable timeframes but will not account for geometric complexities and may not provide realistic estimates of action durations. The converse, attempting to plan by finding satisfying sequences of robot and environment con-

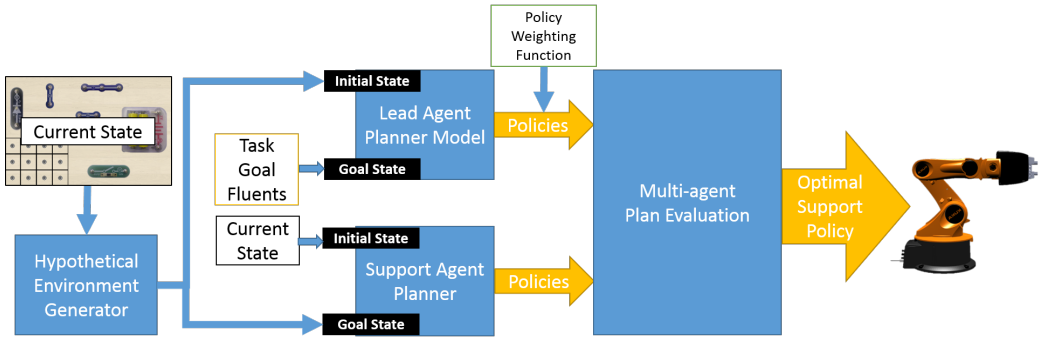

Figure 3.1: A graphical representation of the supportive behavior generation pipeline.

figurations through motion planner sampling and pathfinding, rapidly becomes intractable with problem complexity.

Due to this disconnect between 'possible' and 'efficient' action sequences, seemingly reasonable plans may be far removed from desirable plans both in terms of complexity and resource utilization. Even if motion planners were capable of finding acceptable action sequences without incorporating planning abstractions, this would fundamentally limit the ability of an agent to reason about its task or environment. Many relevant tasks have abstract components that go beyond individual resource usage, and are neatly encapsulated within traditional planning formulations. For example, a goal predicate may specify that a '500Ω resistor' is soldered to 'Wire #1', but it may not matter which particular 500Ω resistor is used to accomplish the goal. As such, it is both desirable and advantageous to seek mechanisms incorporating both types of planner for sequential manipulation tasks.

3.2.4 Supportive Behavior Generation

The goal of the algorithm presented in this section is the autonomous generation of supportive behaviors, with the explicit goal of improving a collaborating agent's performance through the simplification of various aspects of a task via calculated environmental manipulations. The ability to optimize an agent's behaviors for a supportive role broadens its utility, especially within problem domains in which the agent cannot fully perform a specified task either due to liability, lack of capability, or unacceptable risk. Further, the potential reduction in cognitive load attained by incorporating a supportive agent into a problem domain, particularly in the case of a human-robot team, can serve to increase

worker productivity and safety.

Though I am concerned explicitly with teams of agents performing tasks, this contribution makes no assumptions regarding available information transfer between collaborators. As such, the presented algorithm is based upon maximizing the expected performance improvements achieved by supportive actions across a range of possible execution policies. As communicative capabilities are introduced or agents are given sufficient information to model each other's preferences and capabilities, the likelihood of a supportive robot providing increasingly beneficial actions improves.

The algorithm does require the supportive robot to have a model or approximation of the lead collaborator's kinematics, as this is necessary for the purposes of estimating plan completion times given particular environment configurations, which forms the basis of the optimization.

3.2.5 Formalizing the Supportive Behavior Problem

I frame the supportive behavior problem as one of indirect policy optimization, decoupling the support role from the lead agent's planner. Given exclusion criteria such as operators (actions) the supporter is unable to utilize, spatial regions the supporter is not permitted to occupy, objects a supporter is prohibited from manipulating, or predicates the agent is not allowed to effect on the world, a permissible action sequence is generated that results in an environment that minimizes the expected execution time of the lead agents working on the task. Conceptually, this is accomplished by finding the most desirable environmental configurations for the set of likely execution policies being executed by the lead agents, limited by the supportive agent's ability to effect the environment in a way that achieves this ideal environmental configuration. This process is summarized in Figure 3.1.

I define the supportive behavior TAMP (SB-TAMP) problem as the tuple Σ such that

$$\Sigma = (T, \Pi_T, a_s, C_s, s_c, P)$$

where

- $T = \{A, O, C, s_0, s_G\}$ is a TAMP problem such that:

  - A is a set of ('lead') agents
  - O is a set of operators in the form of motor primitive prototypes, representing unparameterized action types
  - C is a capabilities mapping function between agents and operators, indicating actions that may be performed by each agent in A
  - s_0 is the set of predicates precisely specifying the starting environment state
  - s_G is the set of predicates specifying the goal state of the task

- Π_T is a set of symbolic plan solutions for T
- a_s is a supportive agent
- C_s is a mapping function indicating operators from T usable by a_s
- s_c is the current environment state
- P is a set of predicates or partially specified predicate prototypes that indicate prohibited operator parameters

A solution to T is a policy $\pi \in \Pi_T$ that achieves state s_G through a specified sequence of operators in O with geometrically sound parameterizations, executed by a subset of agents in A. A solution to the SB-TAMP problem Σ is a plan π_s which, when executed by a_s, reduces the expected duration for an agent in A of physically executing a plan in Π_T, or reduces the expected search complexity required for agents in A to find solutions $\Pi \subseteq \Pi_T$.

To solve the supportive behavior problem described by Σ, I sample from and reason about alternative environmental configurations, evaluating them based on metrics such as the estimated cognitive (planning complexity) or physical (motion complexity) demands imposed on the agents in the original TAMP problem. This optimization must account for the anticipated time costs associated with achieving the hypothesized improved environment states, as well as the relative likelihoods of a lead agent following particular plans in Π_T.

3.2.5.1 Constructing Hypothetical Environments

At the current environment state s_c I build a set of parameterized operators executable by a_s, which is referred to as O_s. For each action in the support robot's capability set C_s, parameterizations are sampled from a discretization of the task world, and valid instantiations are added to O_s. For a 'pick' operator, the parameterizations of this action include the set of all objects except those whose effects match predicate prototypes specified in P. For example, if the current environment has predicate 'inCircuit(battery$_1$)' and 'inCircuit(*)' is a member of P, a prohibited action parameterization may be 'pick(battery$_1$)' as it would remove battery$_1$ from the circuit. In the case of a 'place' operator, parameterizations include samples drawn from the set of possible legal placement positions and poses in the task area for each eligible object.

With O_s constructed, I then sample different environmental configurations ('hypothetical environments') obtainable by changing individual aspects of the environment. The sampling method I use involves choosing a single object from the scene and moving it to a randomly sampled new location and/or pose. For each hypothesized environment, a plan is determined using actions in O_s enabling the support agent to reconfigure the current environment into the hypothesized state. An estimate is computed for the execution duration of each plan associated with a hypothetical environment (unobtainable configurations are discarded). These duration estimates are later used to evaluate the inconvenience associated with the support robot creating this environment (and tying up the associated resources). Each attainable hypothetical environment is finally encoded as the tuple $\xi = \{\pi, d, s\}$ indicating a plan composed of allowable parameterized operators, an estimate of the duration required to achieve the desired environment, and a set of predicates describing the resulting environment state. I refer to the set of all hypothetical environment tuples as Ξ.

3.2.6 Plan weight determination

I establish a set of plan weights to influence the type of support provided. The selection of this weighting function can have strong effects on the behaviors generated, and as such I characterize three types of weighting schemes that can be used to direct a supportive agent

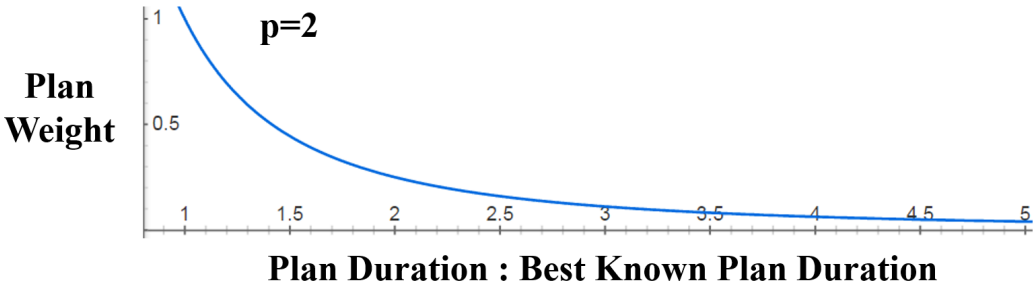

Figure 3.2: Plot of the second weighting scheme enumerated below. This graph shows weights assigned to plan improvements (and impediments) as a function of their optimality versus the best known plan.

toward particular outcomes. To establish these relative weights, I utilize a plan execution duration approximator outlined in Algorithm 5. I define

$$m = \min_{\pi \in \Pi_T} \text{duration}(T, \pi, \emptyset, s_0, f(x) = 1)$$

to be the duration of the shortest (temporally optimal) known plan. Here I describe three cases of useful weighting functions and their resulting behaviors:

1. A conservative optimization function that weights plans relative to their estimated optimality of execution duration. I chose

$$w_\pi = \left(\frac{m}{\text{duration}(T, \pi, \emptyset, s_0, f(x) = 1)} \right)^p$$

for each known plan $\pi \in \Pi_T$, with $p = 2$ (Figure 3.2). Any similar positive function that monotonically decreases as the sub-optimality of a given plan increases can be used to produce analogous results.

2. A greedy optimization function that optimizes for what is estimated to be the best known plan and ignoring consequences for all other possible policies, such as:

$$w_\pi = \begin{cases} 1 & ; \quad \text{duration}(T, \pi, \emptyset, s_0, f(x) = 1) = m \\ 0 & ; \quad \text{otherwise} \end{cases}$$

3. An aggressive optimization function that not only prioritizes making the best plans

better, but also making the worst plans even worse, with the intention of driving a rational agent away from selecting poorly. This is desirable in cases similar to those where a lead agent may either first connect a circuit to a power source or first build the remainder of the circuit. Attaching the power source may increase the danger of the construction operation, and thus makespan, due to the need for increased care in assembly. This can be avoided by the support agent introducing a resource conflict for the power source connection piece, removing it from the available actions the lead may take.

Functionally, this can be accomplished by providing undesirable plans with negative plan weights. It is important to keep the magnitude of negative weights less than the positive weights (using a normalization factor α), or the support robot may perpetually block task progress in the case of partial plan overlap between 'good' and 'bad' plans. A functional example of such a weighting scheme can be achieved by modifying the results from the conservative weighting presented above. I use ϵ to denote a value equal to or slightly greater than m:

$$w_\pi = \begin{cases} w_\pi & ;\ \text{duration}(T, \pi, \emptyset, s_0, f(x) = 1) \leq \epsilon \\ -\alpha w_\pi & ;\ \text{otherwise} \end{cases}$$

3.2.7 Environment state analysis

Combining the information gathered thus far, a supportive agent will choose the best supportive action plan $\xi \in \Xi$ according to:

$$\min_{\xi \in \Xi} \sum_{\pi \in \Pi_T} w_\pi * \text{duration}(T, \pi, \xi, s_c, \gamma)$$

where w_π indicates a plan's associated weight and the duration function computes an estimate of the TAMP solution accounting for the cost of the supportive behavior. The final argument to the duration function, $\gamma : \mathbb{Z}^+ \rightarrow [0, 1]$, is a decay function used to modulate the prioritization of supportive actions causing near-term effects over those that cause longer-term consequences. Providing a function such as $\gamma = \{f(x) = 1\}$ removes this decay

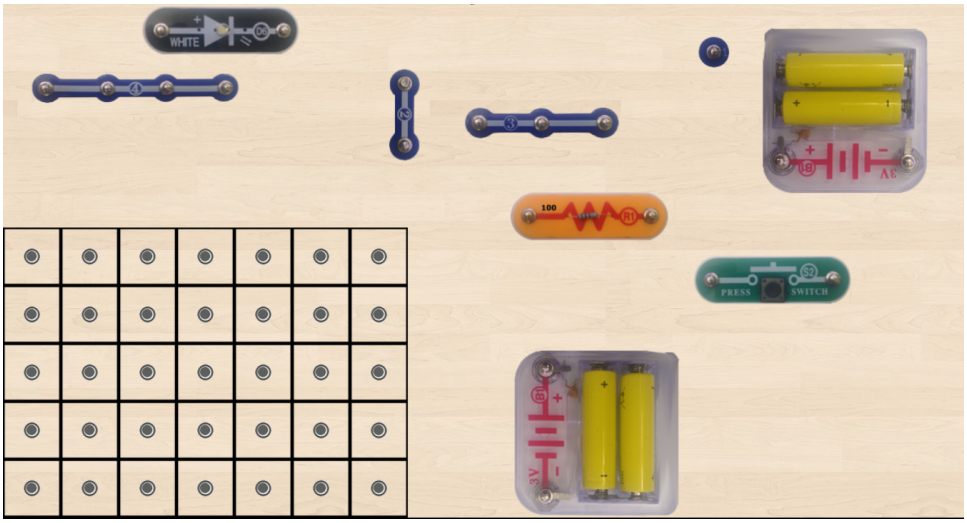

Figure 3.3: One sample scenario used in the evaluation. In this task, the lead agent must create a circuit using two power sources wired in series to drive an LED on a switched circuit. Random configurations of objects were utilized to create a variety of starting conditions.

functionality.

3.2.8 Assumptions and Weaknesses

The method I describe can become arbitrarily computationally expensive depending on the approximations used for the duration estimation function and the size of the known plan set Π_T. As this puts calls to a motion planner in the inner loop of a computation that is highly dependent on the current environment (and challenging to pre-compute), the selection of how coarsely to approximate an agent's kinematics and reachability becomes quite important. As such, it is recommended to initially choose rapidly computable, low-fidelity approximations relevant to the task domain, such as a cost function that computes the shortest, collision-free path for an agent's end-effector to take (independent of its kinematic chain) in the case of tabletop pick-and-place operations.

Once the space of candidate environments and policies is sufficiently reduced, higher fidelity simulations can be utilized, a common technique for expensive evaluations. Directly as a result of this intensive computation in the inner-most loop of the planner, the proposed decoupled approach parallelizes better and converges far faster than an interleaved approach in which the lead and support roles are not decoupled. While this improved performance occurs at the cost of solution optimality, I maintain that the benefits of achieving 'good

Algorithm 5: TAMP Solution Duration Estimation

Input: TAMP Problem T, TAMP solution plan π, Supportive plan tuple ξ, Start state s_c, Decay function γ

Output: Estimated execution duration of π

```
// PLAN_NULLIFICATION_PENALTY is a value greater than the cost of the worst known plan
elapsed_time ← 0;
current_state ← s_c;
foreach i in 1 to |π| do
    Operator o ← π[i];
    if o.preconditions not satisfied by current_state then
        return PLAN_NULLIFICATION_PENALTY;
    step_time ← 0;
    if the set of objects and physical space utilized between o and ξ.π ≠ ∅: then
        step_time ← ξ.d; // Must wait for support plan to finish
    step_time ← step_time + execution_duration(o);
    elapsed_time ← elapsed_time + step_time ∗ γ(i);
    ξ.d = max(0, ξ.d − step_time);
    apply_operator_effects(o, current_state);
return elapsed_time;
```

enough' improvement on a rapidly solvable timescale are far more useful than achieving optimality on a far less rapid timescale.

3.2.9 Evaluation

I performed an evaluation of the supportive behavior algorithm using a collaborative circuit-building task within a simulation environment (Figure 3.3). The circuit-building task is closely modeled after SnapCircuits, a popular children's toy. Using a grid-like board and blocks with electronic components attached to them (Figure 3.4), children use this toy to build simple circuits or devices to learn about electronics. This puzzle is a particularly interesting domain for study as it contains complex, abstract concepts that cannot easily be represented as static literals within operator effect specifications (such as circuit resistance or shorted paths). Additionally, there are non-trivial spatial concerns, as pieces may not be connected side-by-side. Instead, they must be connected via the snap buttons on the blocks (Figure 3.4), resulting in an extra dimension of placement (height) to consider. For example, one could not connect two LEDs in series directly, as a wire piece would be necessary to

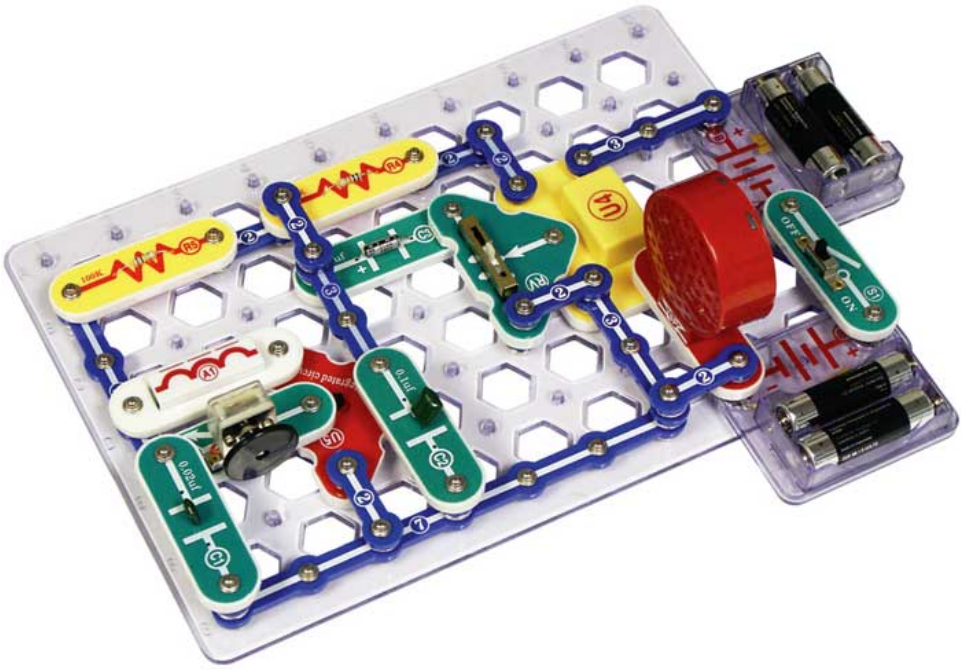

Figure 3.4: Example of a SnapCircuits solution. Snap buttons indicate valid connection points between components, which must be joined to form circuits in a manner that satisfies the implicit 3D spatial constraints imposed by the pieces available and the desired goal circuit function.

bridge them together.

This domain is particularly attractive due to the prevalence of cases where multiple valid solutions exist that differ in terms of their optimality of resource usage. By providing an agent with an excess of SnapCircuit blocks, it is possible to influence it to implement less optimal solutions that are more immediately apparent. I use this case in the evaluation to demonstrate my algorithm's ability to influence an agent away from utilizing these sub-optimal plans, by making more desirable solutions more obvious.

3.2.9.1 Task Environment

I utilize a predefined grasp library, allowing the simulated agents the ability to grasp and place blocks at 90 degree increments. Task execution occurs around a table with full shared reach between the lead agent and supporting agent. Though this reach distance is configurable within the simulation I utilized, I did not add this additional complexity in order to more clearly illustrate the conditions under which my algorithm generates particular

supportive behaviors. The snap board, upon which circuits are constructed, is accessible only to lead agents. Accordingly, support agents are prohibited from executing operators with parameterizations that result in environment effects directly modifying anything on the snap board (specified in the prohibited predicate set $\Sigma.P$).

Collisions are conservatively modeled, maintaining a virtual buffer space around each robot during action execution. In the event of an anticipated collision, the agent with the most priority is permitted to continue its task while the other agent receives a command to return to its home position just off-table. Priority is determined by resource utilization, with the lead agent winning ties. In practice, during conflicts this manifests as the support agent having to place any objects it is carrying on the table, returning to its home position, and then attempting another action. This resolution strategy respects important considerations relevant to real-world collaborative robotics, where robots do not necessarily have the ability to either coordinate behaviors or to explicitly communicate with their collaborators.

3.2.9.2 Scenarios

The evaluation examines two possible worker team scenarios, each differing by the level of knowledge and coordination assumed between the lead and support agent:

The first scenario I consider is that of *unpredictable team behavior*. In this scenario, the lead agent is aware of a random subset of $\Sigma.\Pi_T$. This example has the lead agent following the best valid plan of that random subset with the assistance of a support robot. The support robot attempts to maximize its assistance without specific knowledge about the lead robot's policy, using a balanced plan weighting scheme that favors more optimal solutions (weighting case 1).

The second evaluation concerns *naïve heuristic-driven team behavior*. In this scenario, the lead agent must attempt to solve the task online, aided only by basic planner heuristics that favor plans following a simple prioritization scheme: actions are parameterized using pieces most proximal to the board in their current orientations, before branching out to explore more spatially or rotationally distant alternatives. This agent is also unconcerned with optimality, which may have otherwise affected its planning strategy to be more conservative with the circuit blocks utilized or final configuration achieved. For this example, I

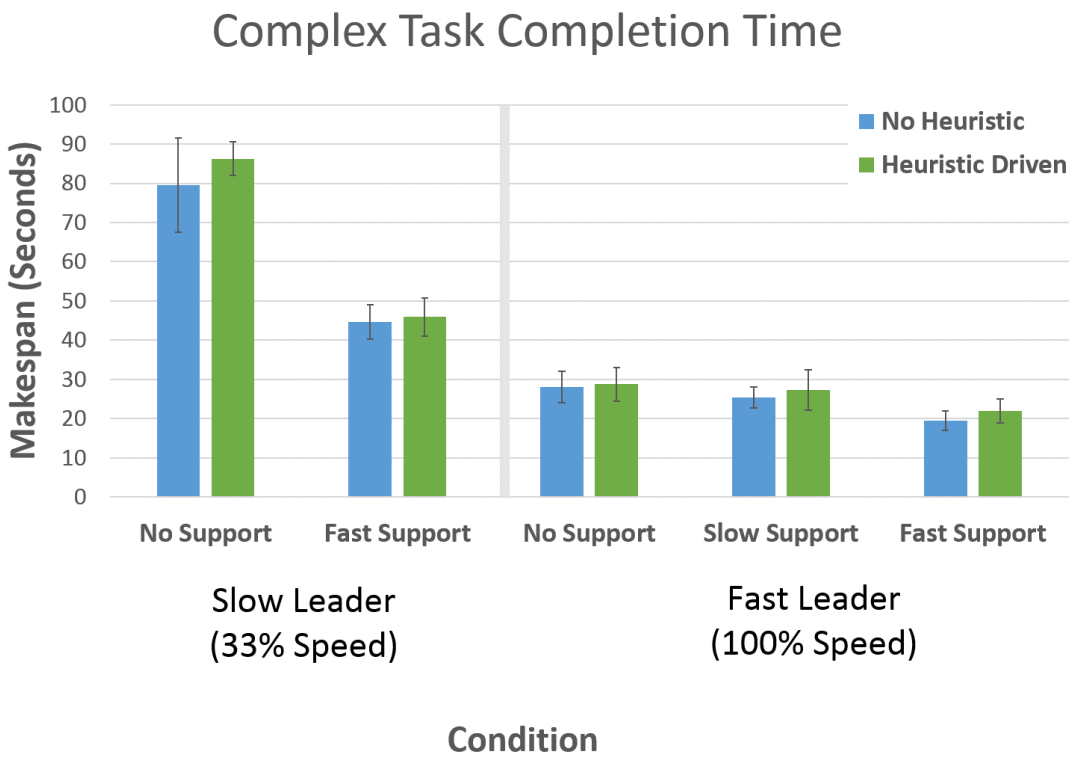

Figure 3.5: Average makespan over multiple random initial environmental configurations of the base task. Adding a support robot provided substantial task completion time improvement even though the supportive agent could not directly contribute towards the task completion.

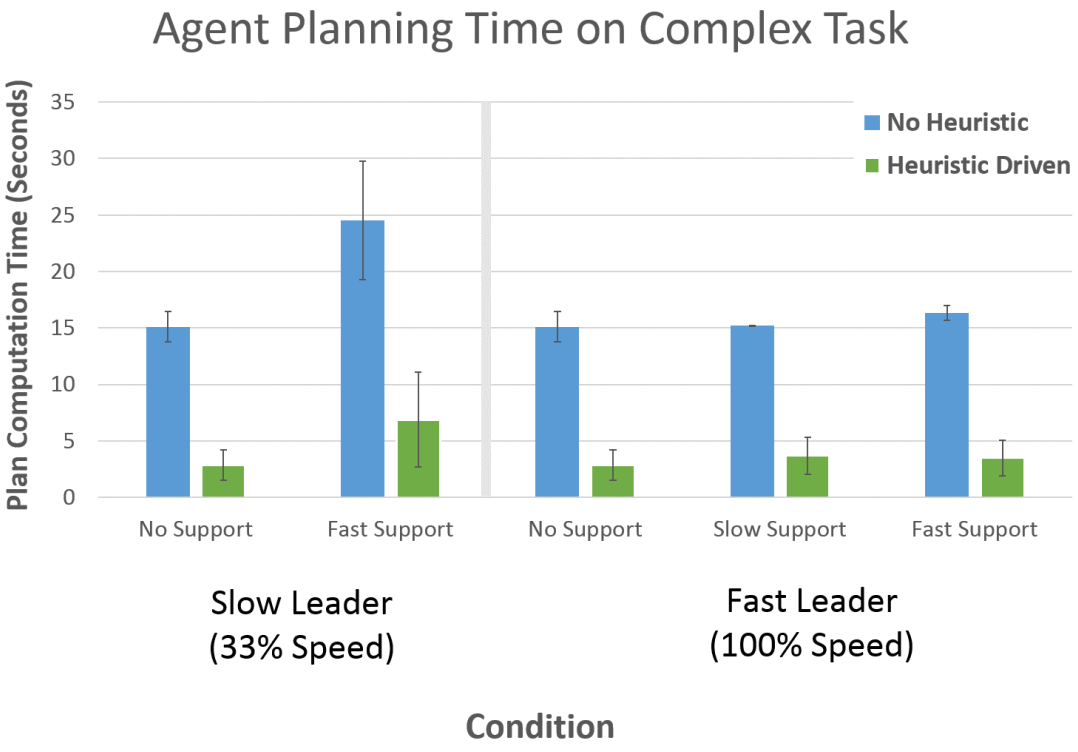

Figure 3.6: Mean planning duration of the lead agent for solving various SnapCircuits TAMP problems. In multi-agent scenarios, due to the decoupled nature of my approach, the agents had to replan when resource or spatial conflicts emerged (as compared to no replanning occurring during the unsupported conditions).

utilize a plan weighting scheme that can actively disincentivize poor plans while driving the lead agent towards more optimal solutions through careful environmental reconfiguration (weighting case 3).

Agent teams were evaluated using lead and support pairs, characterized by their movement speed as being either fast (100% speed) or slow (33% speed). As such, I evaluated support scenarios where a lead agent was paired with either a less, equal, or more rapidly moving helper.

3.2.9.3 Circuit Building Task Results

As shown in figures 3.5 and 3.6, the presented algorithm successfully generated behaviors that reduced the cognitive load and physical effort required by a lead agent solving a complex circuit-building sequential manipulation task. Particularly encouraging for applications into ad-hoc robot teaming is the result from the first scenario, where a knowledgeable support robot and a competent lead robot could successfully collaborate without any direct coordination. The second scenario is especially encouraging for human-robot teaming, where it is possible to leverage the potentially superior symbolic reasoning and planning of a robot while still deriving the substantial benefits attained via the dexterous manipulation and broader contextual awareness of human workers.

Within the makespan (task completion time) evaluation (Figure 3.5), *these results show that supportive agents had helpful effects that drove average task completion times down universally.* I show that a slow lead robot with a fast *supportive-behavior only* agent was able to perform competitively with a single fast lead robot. This improvement demonstrates that through this contribution, a collection of potentially less costly, diversely capable agents can approach or surpass the task completion quality of the less realistic case of a single, universally capable agent.

In this condition, the support robot provided an efficiency increase of 44%, yet did not require the same task-relevant capabilities (precision, tooling, etc.) as the lead. Despite this, its inclusion substantially improved task performance through autonomously generated, off-task, supportive actions. I also see a benefit across task executions for agents paired with slower or equally rapid supportive collaborators, with average time improvements of 9.8%

and 30.5% over the unsupported agent scenario, respectively. These gains were consistent regardless of whether the lead agent was using a planning heuristic known to the support agent. As this task takes place over a small, shared work surface, one can reasonably expect the magnitudes of improvement to scale with task environment size.

I also evaluated the effects of the supportive behavior generation algorithm on a lead agent's ability to rapidly plan high quality solutions for these circuit building tasks (Figure 3.6). This planning time can be seen as analogous to the cognitive load imposed on the lead agent while solving these circuit building tasks. The heuristic driven results I present show that *a supportive agent, aware of a lead agent's planner heuristic(s), can dramatically reduce the search space or planning burden of a task while still maintaining high makespan performance* by autonomously reconfiguring the environment to be more favorable for the thought process of the lead agent. Notably, it was important for the support agent to be faster than the lead as it may have needed to manipulate objects before the lead had a chance to use them. In the case of a lead agent paired with a faster support agent, the 10% average loss in task completion efficiency between the unsupported agent conditions (non-heuristic driven vs. greedy heuristic driven) disappears, while planning speed improvements largely persist. This benefit persists despite my implementation's need to fully replan when conflicts arise. The addition of motion plan caching would further improve these results.

3.2.10 Summary

I conclude this section by summarizing its primary contribution: an algorithm that generates supportive behaviors in ad-hoc multi-agent teams for sequential manipulation tasks. By utilizing an approach that proposes and evaluates possible 'desirable' environment states in the context of potential task execution plans, I present a novel idea that can be used to autonomously create off-goal behaviors that improve robot-robot and potentially human-robot team performance. Alongside the algorithm presentation, I show alternative weighting schemes that can be employed to produce behaviors that can be tailored to particular team dynamics or task considerations. Finally, I present an evaluation of this work within a complex circuit-building domain, showing positive effects on task completion speed and cognitive load.

3.3 Learning from Demonstration and Natural Language

Human-robot teaming advances have the potential to extend applications of autonomous robots well beyond their current, limited roles in factory automation settings. Much of modern robotics remains inapplicable in many domains where tasks are either too complex, beyond modern hardware limitations, too sensitive for non-human completion, or too flexible for static automation practices. In these situations human-robot teaming can be leveraged to improve the efficiency, quality-of-life, and safety of human partners. As a research community, we desire to create collaborative robots that can provide assistance when useful, remove dull or undesirable responsibilities when possible, and assist with dangerous tasks when feasible. In particular, this section focuses on facilitating collaboration through increased team fluency between a lead worker and robotic assistant, complementing prior work that develops collaborative robots as peers with equal authority [5, 11, 119] as well as work on collaborative discourse between humans and robots [49, 141–144].

Whereas the previous section presented a fully autonomous method of generating supportive behaviors, this section addresses the issue that without proper heuristics, such an approach may not rapidly learn and satisfactorily perform optimal (or even helpful) supportive actions for certain types of tasks. In this section I specifically focus on the process of building a system capable of producing an effective robot assistant that learns supportive behaviors from demonstration and verbal interaction. To be effective, this assistant must be capable of learning to anticipate the needs of its co-workers, while maintaining the flexibility to adapt to different worker preferences, retaining value from prior training. In constructing such a system, one must address challenges in state estimation, policy optimization, influencing human task execution, implicit social communication, and policy transfer. Throughout this work, I cover various types of supportive behaviors, ranging from simple stabilization operations to the joint manipulation of unwieldy objects. To ground this contribution within a concrete application domain, I consider scenarios in which a human is performing the joint assembly task of assembling the chair from Figure 1.1b, while sharing a workspace with an assistive robot collaborator.

As prior work has shown that the human-robot interface tends to be the most significant

bottleneck for fluid operation [145], the contribution of this section centers on the development of a mechanism for performing rapid policy learning and transfer for different types of assistive behaviors, with a particular focus on achieving shared expectations regarding when to perform and apply them. To maximize both the applicability of existing research tools and relevance to the broader robotics community, I ground this work within the MDP formalism.

3.3.1 Approach

In this work I consider the challenge of how a robot can account for individual user preferences in terms of execution style and timing in the context of learning supportive behaviors. I treat the supportive behaviors themselves as opaque mechanisms, as it does not matter if they are acquired via traditional reinforcement learning algorithms, TAMP solvers, or learned from demonstration through inverse optimal control algorithms. This contribution is particularly targeted towards those behaviors learned from demonstration, as they may provide benefits not apparent to automated planning methods, thus requiring additional information to guide their appropriate application. I approach the challenge of synchronizing teammate expectations with techniques based upon MDP policy learning and natural language understanding, learning individual preferences from a bootstrapped global policy that is informed by the learned supportive behaviors themselves. A guiding principle for this work is the understanding that the expected conditions under which a supportive behavior *should* execute is typically a subset of the total space of possible conditions under which such a behavior *could* execute.

The remainder of this section concerns techniques for discovering and establishing mutual understanding over what these conditions may be. This is done by integrating supportive behaviors with complex tasks in a manner that leverages semantic models for communicating expectation and intent with collaborators. I begin by outlining a method for associating assistive behaviors with task completion status and the active goals of a lead worker. Once an association can be made between these assistive behaviors and the task at hand, I describe a means of learning assistive behavior preferences from demonstration. I then proceed to specify an approach allowing for the specification and refinement of these

behaviors' initiation and termination states through two-way communication via natural language understanding, allowing for rapid personalization on the level of individual workers, effectively merging natural language understanding with demonstration based methods.

3.3.1.1 Shared mental models

Mutually compatible, joint task comprehension is a crucial element for achieving shared expectation. Prior work has validated the notion that the fluency of a teaming effort relies heavily on mutual understanding and shared expectation, especially within mixed human-robot groups [5]. I represent tasks as hierarchical, directed graphs, encoding both environment states and the actions that transition the world between these states. In particular, I utilize the CC-HTN formulation described previously in Section 2.1. Recall that this representation critically differs from the standard SMDP formulation (where vertices represent environment states and edges are labeled with action choices), as the HTN is built from the conjugate task graph as a means of focusing on agent goals. From the perspective of a robot collaborator, this provides a model of the activities and goals that co-workers are desiring to accomplish. The hierarchical nature of this task representation allows the robot to create associations at useful levels of abstraction (e.g., perform "hold the frame" during the "attach the cushion to the seat" sub-task) as opposed to atomic level actions (e.g., perform "hold the frame" during the "move end effector 1 unit up" action).

By utilizing a goal-centric representation, a robot teammate has a behavioral model of its collaborator within which it can associate assistive behaviors. The active state of the task can be inferred by existing techniques [146] based on state-related observations relating to part positioning, tool use, and workspace utilization. If each vertex of the CC-HTN is intuitively considered to represent a subtask goal for the human (e.g., "drill holes in part 3"), an assistive robot can use this context to decide how best to render help. In the material that follows, I describe a process for learning, applying, and refining these assistive behaviors as continuous controllers that are executed in the context of such a task graph. This is accomplished by learning a distinct policy for assistive actions that can be performed at a given stage in the target task, including the behavior's initiation and termination conditions.

3.3.1.2 Supportive behaviors as continuous motor controllers

It is tempting to consider supportive behaviors as options (temporally bounded discrete actions) [9] whose policies depend solely on the task being performed, though this need not be true. If one considers the task of stabilizing a part so a co-worker can drill holes into it, the act of providing stabilization is a continuous behavior that must begin when expected and (perhaps more importantly) not end until such a time that is in line with the co-worker's expectations. Accordingly, the goal criteria for determining option completion is a function of co-worker expectation rather than some fundamental truth grounded within the observable state space of the task. As such, the policy learning problem for supportive behaviors is more complex than the standard policy learning problem for a lead worker; this is due to the environment dynamics now including the behaviors and (unobservable) expectations of the lead agent. Further, since initiating certain support behaviors may commit an assistive robot for an extended period of time, the agent must be careful during exploration to avoid providing assistance at certain timesteps so that it may learn about actions that would have otherwise been excluded from consideration.

As the leader-follower teaming paradigm imposes an authority structure on collaborators, it becomes necessary to explicitly encode whether any existing supportive behavior controllers are active as features in the lead agents' state space, such that different action policies can be learned depending on the availability of assistance. This can be done in favor of attempting to include state variables fully describing the support agent itself, as this decouples the lead's policy from the particular actions being taken by the supporter (which should be reacting to and accommodating the lead). Doing so removes a circular behavioral dependency that may slow the process of converging to a fluent team policy, as each agent would be optimizing against the other's current behavior, effectively resulting in $\binom{n}{2}$ instances of agents reactively optimizing against each other.

3.3.2 Policy transfer for supportive behaviors

While behavior execution policies can be learned from standard inverse optimal control techniques [2], it is critically important to establish a shared expectation over the initiation and

termination conditions for the execution of such a controller. As I use the MDP formalism to describe agent action policies and world states, this turns the shared expectation problem into one of establishing a commonly understood set of states for each behavior's initiation and termination conditions. In traditional demonstration-based learning, one agent would indicate states that belong to each set either by explicitly annotating them or sampling from them organically via demonstrations during task execution. Unfortunately, this is not a feasible approach for complex tasks, particularly if they have expensive failure modes or are not easily demonstrated without the assistance desired from the supportive robot.

I approach the problem of achieving rapid, transparent policy transfer as one of semantics in both a generative and interpretive framing. Ultimately the problem of synchronizing behavior with expectation is one of converging upon equivalent action distributions over relevant regions of state space for a given task. As reasoning at the individual state level is too cumbersome for communication with humans, a level of abstraction is necessary. Fortunately, there already exists a rich formalism for specifying entire sets of states that lends itself to easily associating (and thus expressing) semantic meaning: classical planning predicates. By using planning predicates as a bridge between high level semantic descriptions and low level sets of environment states, it is possible to connect high level communication to actionable representation, allowing two agents to engage in a dialogue specifying and clarifying relevant regions of state space.

To make the policy transfer more tractable, I factor the task using a CC-HTN, resulting in a significantly reduced state space.

3.3.3 Associating supportive behaviors with subtasks

Each node within the hierarchical goal network contains rich information that can be used to help identify and isolate conditions under which particular supportive behaviors are desired. By specifying individual subgoals within a larger task as being relevant for certain behaviors, learned policies can be automatically associated with identical subgoals at other points in the task or in different tasks with similar subgoals. An example language template for this is "When I am {X}, I want you to {Y}". Within a furniture construction task, this may take the form of the following request: "When I am {attaching the front frame}, I want

you to {give me screws}". This can be extended using conjunctions for either variable, as in the case of: "When I am {attaching the front frame} or {securing the side boards}, I want you to {give me screws} and {hold the rear frame steady}."

In this work's implementation, this association is achieved and confirmed using the bag of words (BoW) approach specified in Algorithm 6. The BoW similarity measure compares the supplied, semantic subtask description from the speech template with a natural language description pre-associated with each subgoal. Once a set of possible subgoals is identified, the agent may communicate these candidates back to the interaction partner, asking for confirmation from the presented choices. Note that in this algorithm, I use a naïve Bayes assumption for the BoW model as it achieves adequate performance and is not the focus of this contribution. Under this condition, choosing the best fitting goal ($g \in G$) according to the provided language ($\lambda_0, ..., \lambda_n \in \Lambda$) is achieved by solving any of the following equivalent statements: $\arg\max_{g \in G} p(g|\Lambda) = \arg\max_{g \in G} p(g)p(\Lambda|g) = \arg\max_{g \in G} p(g) \prod_i p(\lambda_i|g)$.

Algorithm 6: Identify specified subtasks

Input: Hierarchical goal network G, Lexical subtask description Λ,
Number of candidates to return α ($< |G|$)
Output: Subgoal Candidates C

$C \leftarrow []$;
for i *in* $(0, ..., \alpha - 1)$ **do**
    $likelihood \leftarrow \arg\max_{g \in G \setminus keys(C)} p(g) \prod_i p(\lambda_i|g)$;
    $C[g] \leftarrow likelihood$
return C;

3.3.4 Defining initiation and termination conditions

Once the problem space of the task has been factored for a given set of supportive behaviors, it becomes more feasible to learn the appropriate contexts for their execution and termination. In this subsection I present and describe three methods for acquiring these contexts (sets of states): autonomous planning, demonstration-based sampling, and a collaborative discourse-based approach.

3.3.4.1 Autonomous planning

One approach for determining when to begin and when to end the execution of supportive behaviors is through the use of autonomous planners, as done with the Task and Motion Planning contribution from the previous section. While this type of approach can be very effective in determining the space of *possible* initiation and termination states, it is very difficult to account for collaborator expectations and preferences without factoring these features into the state space.

In particular, this method is likely to require a comprehensive understanding of failure modes and the longer-term consequences of each before it is able to evaluate and rank these outcomes against the potential value of the support, an area I identify as future work to extend this thesis. Without a robust model of collaborators and task failure modes, it is preferable to rely on less generally applicable methods in favor of those with stronger guaranteed relevance to the task execution, such as demonstration.

3.3.4.2 Sampling via demonstrations

One effective method of determining appropriate initiation and termination conditions is to learn from actual demonstration during task execution. These conditions can be imparted to a collaborative robot by any number of methods, including imperative voice command, gestural cue, teleoperation, or kinesthetic teaching. The states marked as potential initiation and termination candidates will necessarily be in line with the lead agent's expectations as she will be the agent specifying them. Unfortunately, this method does not work very well for covering wide regions of state space. Since it requires an actual demonstration to learn, the number of available samples will necessarily be low, making it especially difficult to see or generalize observations to large, disjoint regions of state space.

To achieve more coverage on limited examples, one might consider taking the approach of applying Kernel Density Estimation [147] to approximate the confidence that a given state is suitable for initiating or terminating a supportive behavior. While this technique will serve to generalize observations to include nearby states, choosing appropriate bandwidth parameters may be very difficult. Further, for supportive behaviors with irreversible or

dangerous failure modes, such estimation is highly undesirable as there may exist state topologies that have very well-defined boundaries under which catastrophic failures may occur (e.g., an instance that balances an object at the edge of a table, under this method, will provide confidence in states that will likely cause the object to fall). While this method may be coupled with a mechanism that can predict and evaluate failure modes to alleviate these concerns, the development of such a general mechanism remains exceedingly challenging.

An alternative method to demonstration-based sampling of permissible states is providing a means for collaborators to identify them manually. As individually labeling states in this way neither scales to complex problems with large state spaces nor is likely to be easily utilized by non-technical teammates, more powerful, intuitive methods are desirable.

3.3.4.3 Converging expectation and action through dialogue

Language provides a powerful tool and familiar protocol for synchronizing expectations with behaviors. If properly utilized, the abstractions and established conventions of dialogue can be leveraged to rapidly converge on a mutually understood reference to arbitrarily complex, disjoint regions of state space.

Given a lexical description of a state space region, a collaborative robot must properly interpret and confirm its understanding with its teammate to maintain transparency and safety within the interaction. To do so, I provide a template-based algorithm by which a robot can simultaneously interpret state region descriptions and iteratively converge on joint understanding with a collaborator. This is accomplished with the application of two processes that act through parameterized planning predicates as an intermediary: 1) a function that maps language to regions of state space and 2) a function that maps arbitrary regions of state space to descriptive language.

Intuitively, this convergence process depends upon an iterative process of interpreting and then succinctly communicating back descriptors of sets of state space, which may have been identified via any combination of prior knowledge, physical or simulated demonstrations, or language. Given a natural language sentence Λ, planning predicate set P ($|P| = k$), and the Generalized Grounding Graph [148] framework's grounding ($\gamma_i \in \Gamma$) and correspondence variables ($\phi_i \in \Phi$), mapping language to regions of state space involves maximizing

the following probability:

$$\arg\max_{\gamma_1,...,\gamma_N} p(\gamma_1, ..., \gamma_N | \Lambda, P)$$

Factoring this equation into the compositional structure of the given language in Λ, as done in [62], yields:

$$\arg\max_{\gamma_1,...,\gamma_N} \prod_i p(\phi_i | \lambda_i, \gamma_{i_1}, ..., \gamma_{i_k}, P)$$

where the goal is to find the correspondence variables that map most appropriately to individual syntactic components of the provided language. Once the set $\phi_1, ..., \phi_N \in \Phi$ is known, the referenced state region can be specified in disjunctive normal form (DNF) as a disjunction of conjunctions via the predicates indicated by elements of Φ. Each conjunctive clause of the DNF formula can be resolved to a set of states by taking the intersection of its members. Taking the union of each clause reconstructs the lexically specified state set. Given a DNF with disjunctive clauses $c_i \in C$ and conjunctive clauses $c_{i_j} \in c_i$, and a function $R(x)$ that returns the set of all states where the predicate expression specified by x is true, the semantically referenced state space becomes:

$$S = \left\{ \bigcup_{c_i \in C} \left[\bigcap_{c_{i_j} \in c_i} R(c_{i_j}) \right] \right\}$$

The policy for the referenced supportive behavior at these states can then be modified per the intent of the request, such as adding or removing these states from the initiation or termination set of the controller. In collaborative settings, merely understanding language may not be sufficient to guarantee efficient, fluent, or safe interactions. In order to maintain transparency and convey internal state, an agent must be able to express details of its control policy back to its collaborators, effectively revealing its understanding of the team dynamic.

Generating this expressive language involves solving the inverse semantics problem for a set of predicates that best specifies the regions of state space relevant to the policy being communicated. The process of selecting predicates to communicate is equivalent to approximating a modified set cover optimization problem, where the agent wishes to select

the smallest disjunction of predicate conjunctions that covers the desired region of state space. While this problem is known to be NP-hard [149], the ability to use polynomial-time approximate solutions serves to simplify the computational burden. Formally, I wish to specify a DNF formula comprised of predicates that cover regions of state space identified within a supportive behavior's initiation or termination set.

The min-cover problem can be expressed as an integer linear program (ILP), where the goal is to find the smallest cover P_{cover} composed of conjunctions from the power set of available predicates $\mathcal{P}(P)$ such that the target state space region S_{target} is included in the state space regions of predicate conjunctions in P_{cover}. In this formulation, λ is a tunable parameter that can be used to favor conservative (smaller values) or overgeneralized (larger values) covers. $R(p)$ is a function that returns the set of states that satisfies the predicate expression p. $S_{\neg\text{target}}$ represents the set of states that are not relevant to the behavior being characterized.

$$\min \sum_{p \in P} x_p * \left(\lambda |R(p) \setminus S_{\text{target}}| - (1-\lambda)|R(p) \setminus S_{\neg\text{target}}|\right)$$

$$x_p \in \{0,1\} \; \forall p \in P$$

The solution to this ILP can be altered by adding constraints depending on the goals of the interaction. For example, if determining a predicate cover for an initiation set of a behavior where it is critical that all of the states are identified, even at the expense of over-specifying regions, the following constraint may be included:

$$\sum_{p : s \in R(p)} x_p \geq 1 \; \forall s \in S_{\text{target}}$$

Plainly, this specifies that every state must be covered in the solution. If the predicates required to cover the region don't exist, the ILP will be unsolvable, which can be used to

indicate a deficiency in available language. Additionally, the best solution's value can be used as a *direct measure of the expressive power* of the robot's higher level reasoning, and can thus inform interaction designers where additional abstractions are useful.

Solutions to this ILP specify a DNF formula of planning predicates that describe S_{target}. Alternatively, the conversion from predicate set to DNF could be *approximated* via application of a Boolean logic minimization algorithm such as ESPRESSO [150], which was developed for simplifying digital logic circuits. Once a DNF formula of planning predicates has been constructed, these can be mapped to concrete states that the robot can reason with, providing actionable information. Further, with the ability to interpret specifications of state space regions via symbolic predicates, an agent solving the inverse problem can communicate about large regions of state space to others.

Given a region of state space and associated predicate DNF formula that an agent wishes to communicate, simple semantic templates associated with each predicate can be concatenated to form a sentence. For example, the predicate "above(?x,?y)" may be mapped to the semantic template: "?x is above ?y". Continuing the example from Section 3.3.3, a robot could communicate the following using Algorithm 7: "When you are *attaching the front frame*, I can *give screws* when *front frame is present* and *front frame is not attached to left support* or *front frame is present* and *front frame is not attached to right support*".

Importantly, this conversion of understanding to and from the actual state space allows for an agent to explicitly specify implicit constraints that may have been included in the description it was given. Even though an expert task lead may have not included the extra detail in favor of brevity, the ability to explicitly mention implied constraints is valuable in many (particularly safety-critical) scenarios. Further, the precision of the agent's semantic explanation of its initiation or termination states can be quantified, measuring the amount of relevant states that were omitted or the amount of irrelevant states that were included in the DNF formula that was verbalized. This ability to measure the precision of one's communication is helpful for establishing inter-teammate trust and for making uncertainty explicitly known across collaborators.

Algorithm 7: Explain support policy pseudocode

Input: Request for policy explanation Λ, Hierarchical Task Network G, Supportive Behaviors B

Output: String describing requested policies

```
//Identify a request type and respond accordingly.
subtask ← IdentifyReferencedSubtask(Λ, G);
support_behavior ← IdentifyReferencedSupportBehavior(Λ, B);
explanation ← ∅;
if subtask_confidence > support_behavior_confidence then
    //Ex: "How will you help me attach the front frame?"
    explanation ← "When you are {subtask}, I can ";
    behaviors ← All supportive behaviors with initiation states in subtask;
    foreach b ∈ behaviors do
        explanation.append("{b.name} when
          {PredicatesToStrings(SolveForDNF({b.initiation_states ∩
          subtask.states}))}, until
          {PredicatesToStrings(SolveForDNF({b.termination_states ∩
          subtask.states}))})";
else
    //Ex: "When will you hold the rear frame steady?"
    relevant_subtasks ← All subtasks where supportive_behavior has initiation states;
    foreach subtask ∈ relevant_subtasks do
        explanation.append("During {subtask}, I will {support_behavior.name} when
          {PredicatesToStrings(SolveForDNF({support_behavior.initiation_states
          ∩ subtask.states}))}, until
          {PredicatesToStrings(SolveForDNF({support_behavior.termination_states
          ∩ subtask.states}))})";
return explanation;
```

3.3.5 A metric for team fluency

The contribution within this section, aimed toward increasing the mutual understanding of teammates, is presented in support of increasing team fluency. Various metrics have been proposed to evaluate team fluency, such as work by Hoffman [151] which defines it as a measure of the coordinated meshing of joint activity. Fluent teaming achieves smooth meshing of actions through proper timing and execution policies, reactive planning, and rapid adaptation. Others have attempted to use various measures as a proxy for fluency, including makespan, post-task surveys, and teammate idle time. Unfortunately none of these metrics truly or properly capture the fluency of a team, leaving it (unsatisfyingly) an ephemeral concept. Despite this, humans are capable of classifying fluent vs. non-fluent teaming examples, suggesting that this is in fact a measurable quantity. In his work, Hoffman proposes a two dimensional model of fluency, examining either subjective or objective metrics within an observer or participant role. The remainder of this section adopts Hoffman's characterization of fluency and, in proposing a solution to learning supportive behavior policies from interactive dialogue, also proposes an objective, computable measure that can be used to evaluate team fluency as being proportional to divergences between action policy and collaboration expectations.

While fluency is related to efficiency, it is important to disambiguate the two metrics. Task completion time or resource utilization may be a direct measure of efficiency, however fluency can be far more subtle. In this work I propose that fluency is directly correlated with synchronized expectations guided by a theory of mind about one's collaborators; directly informing the ability of an agent to anticipate the plans and contingencies of its collaborators.

Given information about the *how* and *when* of collaborator actions, one can effectively adapt one's own expectations and contingencies to more fluently mesh with teammate behaviors. If one examines agent behaviors within the MDP framework, fluency can be measured as a function of the divergence between an agent's belief state over a collaborator's task policy and the true policy of the collaborator; this holds at both the macro task level (*when* to execute supportive behaviors) as well as the micro level (*how* to execute individual

behaviors).

I define team fluency for a given task within the leader-follower teaming paradigm as a combination of three factors over each possible supportive behavior: the difference between an assistant's initiation states and a leader's expectation of them, an assistant's supportive behavior policy and a leader's expectation of it, and an assistant's termination states and a leader's expectation of them. Divergence between initiation states and expectations for particular behaviors could lead to breakdowns in the form of idling behavior as a leader waits for necessary or desired assistance (e.g., a lead worker waits for a screwdriver that is out of reach to be handed to her, but the assistant doesn't provide it). Divergence in supportive behavior policy and the lead's expectations can lead to task failure or re-planning (e.g., a robot unexpectedly invades the workspace of a human coworker to provide assistance), disrupting task efficiency as well as fluency. Finally, divergence between a leader's expected and follower's actual termination states for a behavior can lead to potentially dangerous task failures (e.g., a robot suddenly lets go of a heavy part that was being jointly carried because it believes it's done helping).

3.3.5.1 Quantifying team fluency

Formally, within the supportive behavior domain I quantify the notion of expected team fluency through policy divergence. In partially observable domains where perfect state information is unavailable or impractical or in continuous domains, the Kullback-Leibler divergence [152] between an agent's *expectations* of others and the *reality* of those agents' beliefs is used to measure policy divergence. In a perfectly observable world with a discrete state representation, one would instead substitute the Wasserstein metric (earth-mover's distance) [153]. Intuitively, performing an optimization over an agent's action policies, initiation states, and termination states that minimizes this proposed metric (to zero) will result in agents converging to behavioral policies where supportive behaviors are initiated and terminated under conservative conditions and actual controller policies will converge to known or anticipated distributions. Mathematically, I quantify team fluency between a task lead L and a set of supportive collaborators C, acting within state space S, with supportive behaviors A as:

$$F(L,C,S,A) =$$
$$\sum_{c\in C}\sum_{a\in A}(\alpha_a+\beta_a+\gamma_a)^{-1}(\alpha_a * D_{KL}(c^a_\pi \| E^L(c^a_\pi))+$$
$$\beta_a * D_{KL}(E^L(p(c^a_{init}))\|p(c^a_{init})) + \gamma_a * D_{KL}(E^L(p(c^a_{term}))\|p(c^a_{term}))) \quad (3.1)$$

where α, β, and γ are behavior specific weights that quantify the potential hazard in a teammate mischaracterizing each of the behavior's components; namely the execution policy, starting conditions, and termination conditions. For example, in a collaborative carrying scenario there is far more danger inherent to misunderstanding the termination conditions than the initiation conditions. Likewise, for a robot that is performing a spot weld, understanding the initiation conditions for the behavior may be far more critical from a safety perspective than understanding the execution policy itself. In this measure, a lead collaborator L quantifies its expected fluency with its teammates, where $E^L(c_{\pi_a})$ denotes L's expectation of c's action policy π for support behavior a.

For particularly low-risk domains, it may be desirable to instead utilize the reverse formulation $D_{KL}(p(c^a_{init})\|E(p(c^a_{init})))$ and $D_{KL}(p(c^a_{term})\|E(p(c^a_{term})))$ to drive an approximation of the initiation and termination sets, as this will provide an approximation that is more inclusive (potentially including regions of state space that should not be), covering more of the breadth of the true distribution.

3.4 Social Cues for Task Understanding and Role Evolution

One of the hallmarks of development is the transition of an agent from novice learner to able partner to experienced instructor. While most machine learning approaches focus on the first transition, this section deals in building effective learning and development mechanisms that allow for the complete range of transitions to occur. In this section, I present a mechanism enabling such transitions within the context of collaborative social tasks. This cooperative robot is capable of learning about task execution from an experienced human user, collaborating safely with a knowledgeable human peer, and instructing a novice user

based on the explicit inclusion of a feature based on projected spatial occupancy within the system's planning and skill execution subcomponents. I conclude with an evaluation of this feature's flexibility within a collaborative construction task, changing a robot's behaviors between student, peer, and instructor through simple manipulations of this feature's treatment within the planning subsystem.

Most robotics learning systems focus on developing a robot to be either a student or a teacher. In robot-learner applications, a robot is tasked with acquiring a skill from some collection of input signals, examples of which include human demonstrations, simulations, or solutions to complex sets of constraints. In robot-instructor applications, the robot focuses on transferring a set of knowledge to another agent, measuring competency and modulating the pace of knowledge transfer. These two domains are often treated as unrelated, with little crossover between communities. Between the two exists the relatively new field of socially collaborative robotics, a distinct but related research domain focusing on human-robot teaming and interactions. Collaborative robotics is interested in robots as capable learners, peers, and instructors, examining both the roles themselves and transitions between them. As robots transition out of isolation into roles where collaboration with humans or other robots is possible or even required, a focused effort must be made to engineer systems that can accommodate these complex requirements.

A multitude of challenging research problems must be addressed to enable fluent interactions between humans and robot agents [123]. Facilitating communication between humans and robots constitutes a substantial portion of these issues. Human teams constantly communicate explicitly and implicitly over a variety of verbal and non-verbal channels. Humans are capable of leveraging complex intention recognition capabilities in real-time, developing and updating models of their co-workers as they gain experience collaborating with or observing them. Further, we are able to infer a great deal about complex interactions between others merely through observations. This holds true at a distance when the actors are reduced to low fidelity representations of themselves, even in the absence of explicit context. While intention is tremendously complicated, interpreting basic motion is not, requiring very little from one's environment to provide context. This irrepressible, constantly utilized ability to infer intention from motion develops quickly in children around 9 months

of age [154]. The limited processing requirement combined with the vast potential for understanding agent behaviors makes this an extremely attractive phenomenon to interpret and incorporate into collaborative robot systems.

Though people share a host of available avenues for communication, synchronizing mental models remains a challenging task for human teams. It has been demonstrated that increased implicit communication directly improves the performance of teams under stresses caused by temporal constraints or uncertainty, by enabling members to act in anticipation of teammates' actions [65]. To build robot systems capable of natural collaboration with humans, a targeted engineering effort and planned interaction methodology must be explored and developed regarding intentionality detection and expression. Additionally, integrating teammate-like behaviors into collaborative systems may dissuade notions of robots being unsuitable or unready for certain application domains.

Beyond the difficulties inherent to developing cooperative systems, complexities within skill and task representation further complicate the design and production of socially capable, collaborative robots. While humans often have little trouble verbalizing their understanding of a skill or task, robot systems' internal representations often do not afford such transparency. Designers must architect socially oriented solutions to the problems of skill acquisition, role definition, role selection, and action selection. Hierarchical Learning (HL) has yielded substantial success relating to the challenges of task representation, allowing for the scaling of skill complexity by dividing intricate tasks into collections of simpler, related sub-skills [37]. Concepts from HL have also been leveraged within individual skill acquisition to improve learning from demonstration and skill transfer between similar environments [39]. The weakness inherent to such HL techniques is the lack of explicit specification for human-accessible representations of information.

Addressing the challenges inherent to skill acquisition, task representation, and role selection introduced by building a collaborative system requires explicitly architected, socially directed features within these core behaviors. I define Social Hierarchical Learning (SHL) as an extension to HL designed to facilitate the development of systems that can flexibly acquire skills from multiple sources, generalize these skills to cooperative tasks, and execute a complex plan collaboratively as part of a mixed human-robot team while maintaining

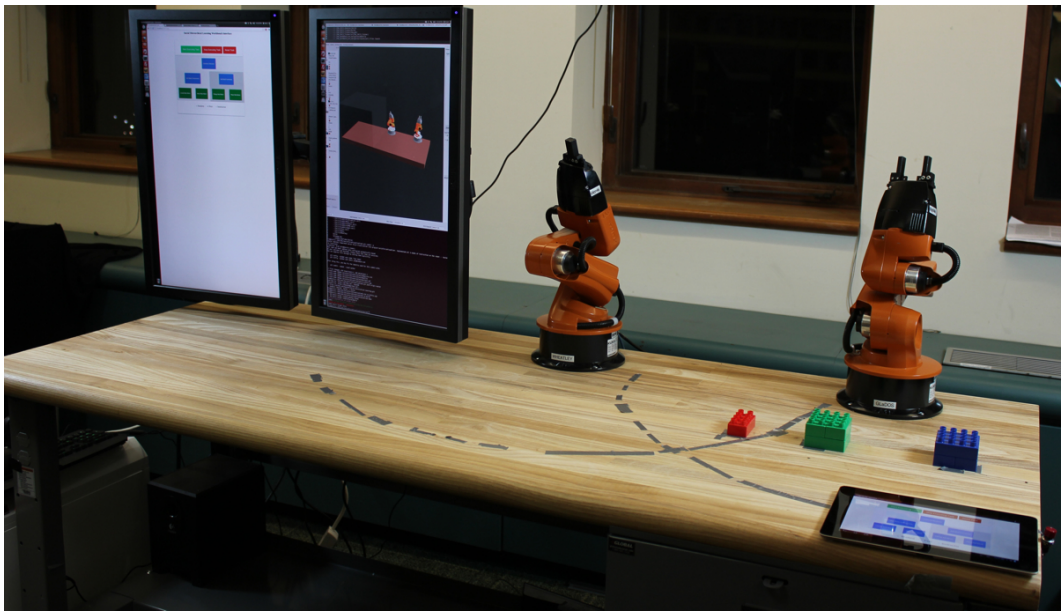

Figure 3.7: The Collaborative Workbench used in the proof-of-concept implementation. The left monitor mirrors the tablet's display, showing live information about the task tree throughout task execution and allowing human participants to claim roles within the system. The right monitor displays a simulated environment mirroring the real world, along with social force fields as they are generated from user motion.

transparency to its peers.

Within this section I examine the utility of a feature described as *social force*, a projection of external agents' anticipated movements and spatial occupancy. Social force can carry vastly different meanings depending on the context in which it is applied. As an example of social force expression, consider two humans A and B selecting objects from a table. If A and B simultaneously reach for an object from the same region, the reaching action of A may cause B to select objects from a different region. The likelihood of a change in B's selection region depends on a number of factors, including the magnitude of repulsive social force expressed by A's action and B's commitment to the initially selected area. However, if A points towards an object and communicates intent for B to select it, the magnitude of the attractive social force expressed by A's gesture (modulated by insistence/sharpness of motion) and B's commitment to the initial selection will determine if B follows A's suggestion. I have integrated this feature into the skill execution and planning subsystems of a kinesthetically trainable, collaborative robotic workbench with two robot arms.

3.4.1 Related Work

Skill acquisition within state of the art robotics systems can be accomplished through a variety of methods ranging from methods accessible to true novice users with no robotics or programming experience to methods requiring expert roboticists intricately familiar with the hardware, software, and sensor suites available to the robot. Techniques such as learning from demonstration (LfD) allow subject matter experts who may be novice robotics users to impart knowledge and abilities into robot systems [14]. These methods work exceptionally well for collaborative applications, as the robot's teammates may not have robotics or programming expertise. LfD is used frequently for human subjects experiments, providing valuable feedback to interaction designers [16–18]. Within LfD, kinesthetic teaching has shown exceptional utility as a method of learning skills from novices [92, 155–157]. By physically guiding a robot to perform a skill, users are capable of directly imparting knowledge to a robot through example skill executions driven via a physical, intuitive interface. Other advantages of this method include no requirement of external sensing equipment, no danger of violating kinematic limitations of the robot, and minimal danger of unexpected self-collisions [19].

Once individual skills are learned, one must focus on task-level completion. Fluency of collaboration can only be attained once members of a team have a consistent mental model of the task to be completed. This mental model must include information relating orderings of skills, the mechanics of each skill's execution, and skill groupings that make up roles that can be assumed. Hierarchical task representation offers a convenient way to condense large quantities of information, exposing the proper level of sophistication required for the requesting agent or process. There exist several algorithms capable of decomposing tasks into representations suited for autonomous agents [37–40, 79, 158], but the addition of human co-workers introduces requirements for human-readable and human-communicable task representations [41, 42]. Recent contributions in this area have yielded human-inspired methods of action segmentation [43]. Statistical regularities in observed action sequences can be used to discern meaningful segmentations by identifying boundaries between probable action groupings. Other means of introducing shared mental models

include exposing an interface by which a human can share his or her mental model of the task with a robot, overriding any existing representation [50–52]. Converging towards common mental representations of tasks represents significant progress towards fluent human-robot teaming, and can potentially simplify the task of interpreting the intentions broadcast by one's peers.

Once a common model of the task's components has been established, the next challenge facing a team lies in learning individual members' preferences for execution. Given potentially parallel branches of subtasks, it is important that teammates have accurate models of each other's expectations. Adaptive planning systems capable of modeling human behaviors with social understanding are beginning to emerge and face testing in real-world environments. Much active work remains in both the area of socially-oriented adaptive planners and that of addressing unfulfilled team commitments or execution-time variation in ordering constraint fulfillment [77, 78]. One effective method of learning teammate preferences is cross-training, a technique in human-human training in which participants trade roles. This approach has shown some promise when applied to human-robot teams, improving both the comfort of the human operator and the performance of the team [5].

One particularly promising avenue of exploration for building solutions to these collaboration-centric problems is intention detection and classification. In this domain, the unseen social forces that agents exert upon one another have been shown capable of yielding rich social, cultural, and intentional information [159–161]. It's been shown that details about inter-agent relationships and causal relationships between agents and the world can be deduced from observations of impoverished and low-context trajectory data [162]. While social forces have been used in the past within perceptual tasks both to understand the actions of agents as intentional and to classify agency, I integrate social force understanding into a collaborative robot's planner to provide it the ability to dynamically transition between the roles of student, collaborator, and teacher.

3.4.2 Approach

The described system is implemented as a minimalist component of a larger SHL system, seeking to explore the addition of socially oriented algorithms to traditional Hierarchical

Learning. The primary motivations behind SHL systems are to enable flexible skill acquisition from non-experts, generalize skills to social, cooperative tasks, and be able to collaboratively execute these skills within the context of complex tasks as part of a mixed human-robot team. I am also interested in facilitating bi-directional skill transfer. Many learning systems primarily focus on methods of inputting skills, but truly collaborative systems must also be able to effectively communicate learned knowledge to other agents.

I have implemented this collaborative task execution proof-of-concept on the Collaborative Workbench platform (Figure 3.7), a 1.83m x 0.762m worktable equipped with two KUKA YouBot 5-DoF light manufacturing arms spaced 0.5m apart, with approximately 0.5m of overlapping operational envelope. Each arm is equipped with a 2-DoF gripper. Sensing is performed through registered point cloud data, captured via a Microsoft Kinect mounted above the workspace.

The collaborative system that was implemented to demonstrate social force is built in the Robot Operating System (ROS) software framework. Core data structures, such as skills, task trees, and robot controller interfaces are incorporated into a static software library. Other necessary, live components such as the social force publishing service, kinesthetic skill trainer, hand tracker, and cooperative execution program are implemented as separate ROS nodes. Following this type of module-oriented design pattern within ROS allows for near-trivial portability between application domains. User interfaces are implemented in HTML and Javascript, utilizing the ROSBridge web interface to allow for device portability, effective visualization, and rapid prototyping.

3.4.2.1 Kinesthetic Teaching

Skill acquisition within my system is accomplished via demonstration, through kinesthetic teaching. Kinesthetic teaching is a process by which a user physically manipulates the learner through an execution of the target action. While there are many valid ways of performing kinesthetic teaching, each with various strengths and weaknesses depending on the target domain, the presented system is based upon keyframing as opposed to pure trajectory-based recording. Keyframing is a process by which actions are recorded as a sparse set of important animation frames. Playing keyframes back in sequence can be used

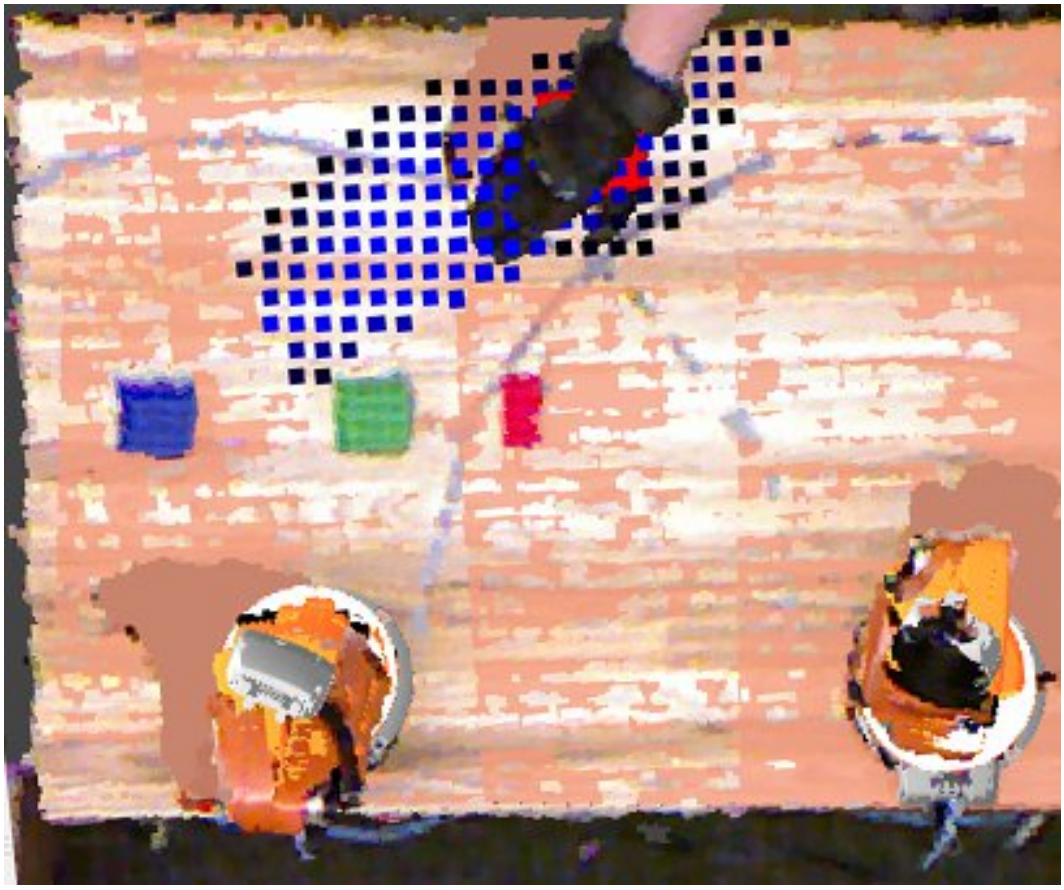

Figure 3.8: Realtime visualization of a social force particle field generated from the user's motion patterns, rendered over point cloud data from the workbench sensors. Social force magnitude is represented by the blue channel, with the intensity of force increasing as particles range from black to blue. The estimated position of the user's hand is represented by a red square, shown under the gloved hand.

to recreate the skill as demonstrated. Individual trained skills within SHL systems are intended to be primitive action components that may be combined to achieve high-level functionality.

To train a skill within this system, a user must explicitly engage the robot into training mode through the provided web interface. Once training has started, the user is free to pose the robot to the desired position and set keyframes as necessary. Features to be tracked (e.g., motor positions, gripper distance to object, etc.) are indicated by the user prior to training, rather than learned during training. Keyframed state feature vectors are recorded as vertices within a skill's state graph, initially linked by edges reinforced from the explicit training example. As skills are executed, more densely populated paths are created, as

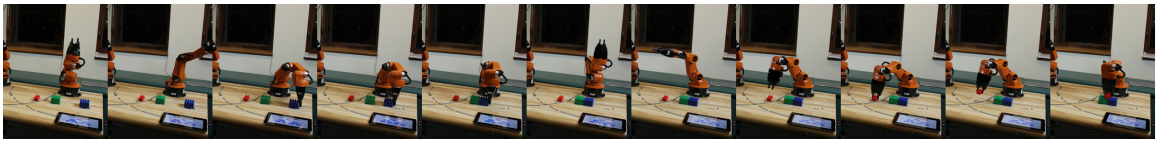

Figure 3.9: Full demonstration of the task by the robotic agent. When multiple agents participate, it is possible to execute the subtask that joins the blue and green bases and the subtask that moves the red block into position in parallel.

explored states are recorded and linked. I explicitly simplify this component of the system to retain the minimally viable interactive feature subset to illustrate the flexibility and magnitude of effect of social force as a feature. Additionally, only a linear interpolation exploration function was used during skill execution, leaving the explicit user-trained paths as the only available options for the system to follow.

3.4.3 Task Structure and Execution

Complex tasks are represented as a hierarchical collection of primitive skills with varying goals and pre-/post-conditions. A task tree is a generalized representation of a complex task sequence. Branches represent divisions of labor along potentially parallel paths of execution, each relating to a possible role to be fulfilled by a participating agent. As tree nodes are claimed by human or robot agents for execution, all child nodes are also claimed for the agent. Ownership of subtasks can be released, returning the unfinished and newly unclaimed skills back to the available work pool. The task tree is managed by a process independent from working agents and is responsible for managing ownership requests and status updates. A web interface provides users with a visualization of the task tree, including information such as branch ownership and completion status.

The collaborative execution of a task is accomplished through agents claiming ownership of subtasks from the task tree and executing the associated skills. It is the responsibility of each agent's planner to determine which subtasks to request ownership of and fulfill. For my tests, I utilized a simple planning algorithm that would choose the most promising role according to a social force value associated with each subtask at the time of the decision, fulfill it, and query the task tree for additional roles. An agent only claimed one branch of subtasks at a time. In the absence of social force measurements or in case of ties, decisions

would be made randomly amongst candidate options. Throughout execution, agents would evaluate unclaimed roles against their chosen role, aborting their choice if the difference in desirability (measured solely through social force) between an unclaimed role and the currently assumed role passed a predetermined threshold. If no roles are unclaimed and the chosen role is evaluated to have low desirability beneath a predetermined threshold, the agent would halt its actions, continuing once the role's desirability increased above the threshold again.

Enabling collaboration between humans and robots is a major research challenge within the robotics community. Collaborative robots must be capable of learning co-worker role selection preferences in addition to modeling optimal role selection choices for themselves. Scenarios where multiple agents interact within a shared environment increase the likelihood of collisions between workers, both in terms of physical space occupancy and resource requirements. Collaborators must also be able to adapt to new situations, ranging from changes in skills required to perform one's duties to training new teammates. These high-level abilities all require some form of intention modeling to be effective.

One feature that I demonstrate as capable of assisting in fulfilling these requirements is that of social force. I define social force as a projection of one's intention through movement. While physical force generates straightforward behavior when applied to most agents, social force affects each agent differently and can be context-sensitive depending on the skill being executed. The social force exerted by an agent at a point in space can be computed given a model of the agent along with its motion history. In the absence of a kinematic model, the end effector position in space may be used.

3.4.4 Computing Social Force

Given a series of samples S of the target agent's end effector positions, one can compute an ellipsoid representative of likely future positions of the agent as a projection of anticipated motion. For work performed on the Collaborative Workbench, I used a 2D projection of the samples (Figure 3.8), as including the z-axis did not offer any accuracy or responsiveness gains for its computational cost. I define a temporally bounded set of positions $P = \{p : p \in S_{[t_{now}-t_{duration},t_{now}]}\}$. For these experiments, I empirically set $t_{duration} = 0.75s$. Adjusting

$t_{duration}$ changes the sensitivity of the social force projection, with increasing values softening the effect of sharp motions and decreasing values increasing their effect.

I obtain the covariance matrix of the samples

$$M = \begin{bmatrix} cov(P_x, P_x) & cov(P_x, P_y) \\ cov(P_y, P_x) & cov(P_y, P_y) \end{bmatrix}$$

which is guaranteed to be a real, symmetric matrix (for any dimensionality sample set). Given this guarantee, I can obtain real, orthogonal eigenvectors describing the principal components of P through the eigendecomposition $D = VMV^T$, extracting eigenvectors from columns of V and eigenvalues from the diagonal matrix M. Using the eigenvectors as axes and eigenvalues as axis lengths, I construct an ellipse modeling the second order moments of the sample, providing a rough description of the data's shape and orientation while remaining robust against outliers from sensor fluctuations. Finally, I translate the ellipse's center by a value c representative of the inertia of the latest sampled position $p \in P$, given by $c = p - \bar{P}$.

Once the social force ellipse is defined, determining the social force value for a given point is a fast operation. A transformation matrix T can be constructed from scaling, rotating, then translating the unit circle into the ellipse described by the parameters extracted from the steps above. Given a test point x, one need only apply the transformation $x' = T^{-1}x$ and test for the presence of x' within the unit circle. The distance of x' from the origin along each axis may be used to scale the social force value assigned to x.

3.4.5 Role Transitions Leveraging Social Force

Simple changes in treatment of social force within a robot's action planning algorithm can be used to dramatically alter its behavior between that of student, peer, and instructor. The robot can use generated social force fields to evaluate potential skill or role choices. By obtaining readings of social force at the various locations its end effector will travel through at keyframes within the candidate skills, the robot can use social force to affect its skill selection decisions.

To illustrate the effectiveness of social force as a transformative feature within human-robot collaboration, I use a simple construction task as a test domain (Figure 3.9). Three blocks are set on the worktable, a red top piece, a green base piece, and a blue base piece. From the initial setup, the goal is to join the two base pieces with a single top piece in the middle of the working area. The task tree consists of two branches, one to move the blue base next to the green base and one to place the red piece on top of the joined bases. Each branch consists of three subtasks: a 'grasp' or 'locate' action, a 'place' action, and a 'return to resting pose' action. The 'return to resting pose' action is not displayed in the task tree visualization figures. The first actions of each branch can be done in parallel, but the red piece cannot be placed until the base pieces are moved together. This construction task constitutes the simplest possible task formulation that allows for meaningful expression of all modes of operation that I wish to demonstrate.

3.4.5.1 Robot as Student

If social force from a particular agent is treated as an attractive, positive force within the action planning system, the robot will behave as a learner, as if taking cues from an instructor (Figure 3.10). As the designated instructor gestures towards particular areas of the workspace, the generated social force will positively weight any decisions involving skills passing through that space. By enforcing a minimum, non-zero value of social force required before making a role or skill choice, one can ensure that the robot will remain inactive unless instructed towards a particular choice. The order of actions taken from this behavior can be used to learn task-level preferred action orderings that may not be obvious given data inherent to the skills in question (e.g., it's typically better to pour the cereal into the bowl before the milk).

During the construction task, the robot evaluates its skill choices within the task tree and chooses the skill that best fits the working area covered by the social force of the instructor. As the instructor motions towards the blue base block, the robot performs its locate blue block skill, moving its end effector to the side of the block, ready to push it. Once the instructor gestures towards the green block, the robot executes its push blue block skill, moving the block to its desired position. Similarly, if the instructor gestures towards

the red block first, the robot will execute its pick up red block skill. Once the instructor moves the blue block to its final position, social force exerted near the newly combined base region will trigger the robot to execute its place red block action.

3.4.5.2 Robot as Collaborator

When social force is treated as a negative, repulsive force within the action planning system, the robot will behave as a peer, choosing subtasks that can be fulfilled in parallel with other agents while minimizing collisions (Figure 3.11). As other agents begin to execute their roles, the robot will initially choose from the minimally conflicting options. During execution, the robot becomes capable of reacting to invasions to its workspace, choosing to adopt a different role when it becomes apparent through other agents' social forces that it will be interrupted.

Throughout the construction task, the robot chooses to execute parallel, non-conflicting subtasks in concert with the agent it is working with. As an agent begins to perform an action, the robot is biased against choosing spatially similar actions to execute. If a robot has already chosen a particular task, but detects negative social force due to a conflicting task choice by another agent (e.g., human co-worker) operating in the same space, the robot aborts its selection and chooses a different, less conflicting action. This behavior may also lead to instances of turn-taking, which while typically avoided in collaborative exercises, can lead to safer operation in confined workspaces.

3.4.5.3 Robot as Instructor

If social force is used as a trigger once its value passes a particular threshold, the robot can behave as an instructor, indicating which skills it desires its student to complete and in which order (Figure 3.12). Leveraging the keyframing within the trained skills, a robotic instructor can step through the skill in stages until the social force trigger criterion is met. More formally, given a skill A consisting of keyframes $\mathrm{kf}_1, \mathrm{kf}_2, ..., \mathrm{kf}_n \in A$, the robot will execute keyframes in the set $K = \{\mathrm{kf}_i \in A : i \leq (\#\text{ times demonstration has been repeated}), \mathrm{kf}_i \notin \text{GoalStates}(A)\}$. The robot first executes keyframes $k \in K$ in increasing order from $k_1, k_2, .., k_{|K|}$, then reverts to its start state by executing keyframes $k \in K$ in reverse order

from $k_{|K|}, k_{|K|-1}, ..., k_1$. This staged execution has the benefit of progressively revealing more of the desired action while leveraging existing information to do so. Once the student begins to perform the desired action, the trigger criterion will be met and the robot is able to move on to the next desired skill.

When social force is treated as a triggering feature during the construction task, it can be leveraged to teach skills or orderings to the robot. Using the staged skill execution technique, the robot executes instructive behaviors from existing skill data. The robot chooses its preferred ordering of tasks to execute and directs another agent to perform them. In the construction task, the robot was trained to instruct the user to perform the following ordering: locate blue block, move blue block, pick up red block, place red block. While waiting for the user to indicate knowledge of the desired action and to demonstrate the intention to perform it, the robot begins to demonstrate the introductory motions of the skill. This demonstration is performed in increasingly complete motions, stopping just before completing the goal of the subtask. Within the context of the pick up red block skill, the robot first moves above the red block. If the user does not respond, the robot then begins again by moving over the red block, opening and orienting its gripper such that the block can be grasped. Finally, if there is no response from the user, the robot executes the entirety of the action with the exception of the final goal state: moving over the red block, orienting its gripper towards the block, opening its gripper, and placing the red block within its fingers before aborting and returning to a resting position.

With minor changes in treatment, social force is a feature that makes collaborative robotic systems capable of transitioning between student, peer, and instructor roles. With minor changes to the robot's skill execution algorithm, the robot becomes capable of teaching other agents kinematically demonstrated actions, in addition to teaching task ordering preferences.

3.4.6 Summary

In this section I introduced a rich feature called social force to a collaborative robot's planning and skill execution subsystems. This feature can be calculated in real-time and has been shown capable of providing robotic systems a range of social, collaborative function-

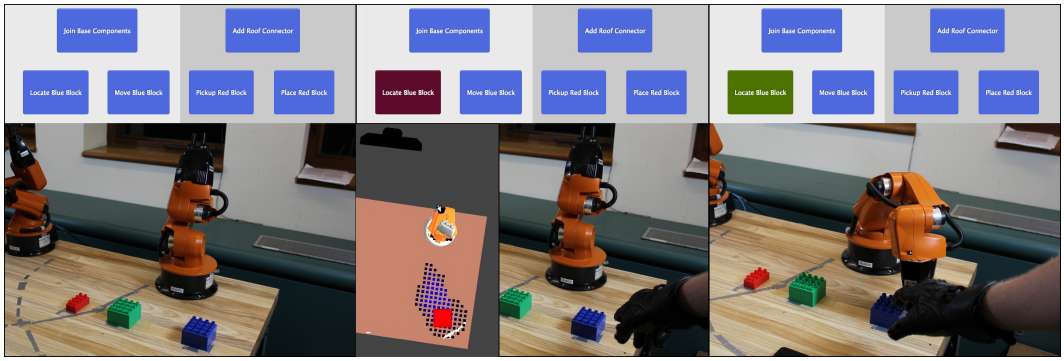

Figure 3.10: An example of the "student" behavior, achieved by attractive social force. In the first panel, the robot is waiting for guidance before choosing an action. In the second panel, the user has gestured near the blue block, exerting social force in the goal region of "Locate Blue Block". The red box around the skill name indicates that robot intends to complete this action. The final panel shows the action as 'completed' in the task tree visualization (green box).

ality based on its treatment. By adding social force considerations into a robotic system's planner, the robot becomes capable of working with others as a peer as well as learning task ordering from others as a student. With minor changes to a robot's skill execution algorithm, it becomes capable of using social force to teach skills and task orderings to others. I've demonstrated this functionality through a proof-of-concept implementation on a collaborative workbench equipped with a robotic, lightweight manufacturing arm. The advantages offered by this approach can result in improved team efficiency and robot integration into social roles.

Incorporating social force as a feature within a collaborative planner and skill execution algorithm yields promising results, meriting further study through inclusion in more complex, more capable collaborative systems. Future work exploring the benefits of this feature includes correlating the robot's own projected social force values with environmental changes that occur during skill execution to learn about the consequences of one's action paths. Agents can also learn models of collaborators' social force tolerances, and use this data to optimally adapt skill execution to minimize cross-agent conflicts. Socially collaborative robots can also correlate social force values with collaborators' reactions to determine whether an agent is looking to occupy a student, peer, or instructor role when interacting with the robot.

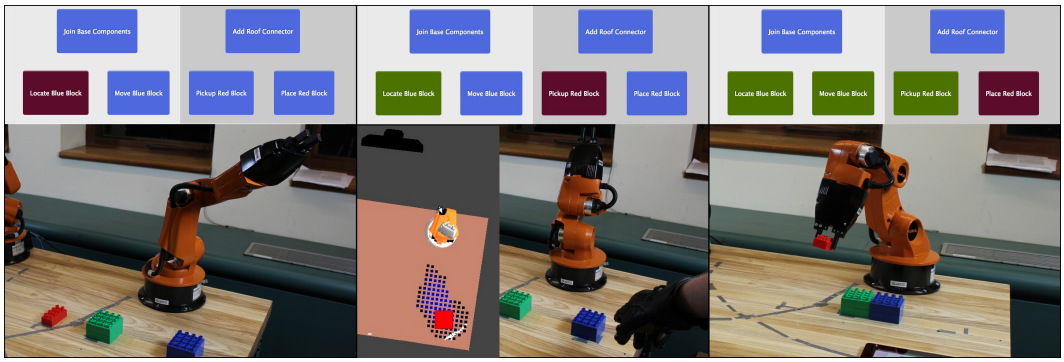

Figure 3.11: An example of the "peer" behavior, achieved by repulsive social force. In the first panel, the robot intends to complete "Locate Blue Block". The second panel shows an interruption of the action by a human user. Upon detecting strong social force in its current skill's goal region, the robot aborts its execution and chooses a different, non-conflicting skill. The third panel shows the robot continuing to exercise conflict-avoidant behavior with the user.

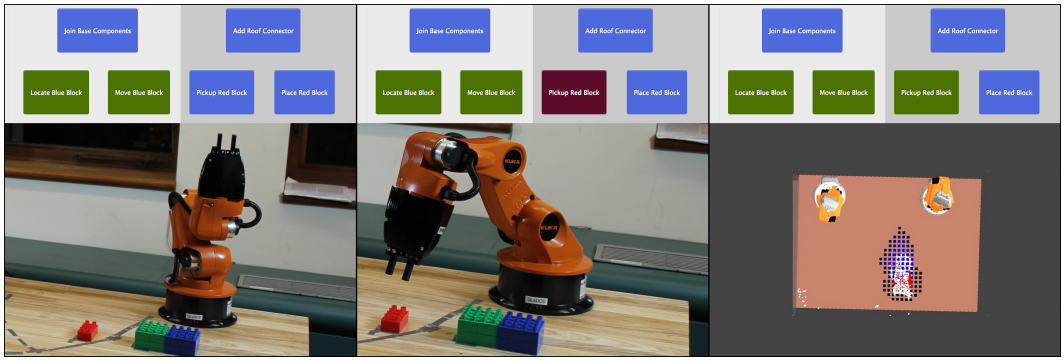

Figure 3.12: An example of the "instructor" behavior, achieved by a using social force as a trigger. The first panel shows the robot evaluating available actions to teach. The second panel shows the robot demonstrating part of the "Pickup Red Block" subtask without actually completing it. In the third panel, the user has picked up the red block as previously pointed to by the robot, triggering the robot to mark the task as complete.

3.5 Enhancing Agent Safety and Learning through Supportive Behaviors

Exploration and self-directed learning are valuable components of early development. This often comes at an unacceptable safety trade-off, as inexperienced agents (such as infants and toddlers) are especially at risk from environmental hazards that may fundamentally limit their ability to interact with and explore their environments. In this section I address this risk through the incorporation of a caregiver robot, and present a model allowing it to autonomously adapt its environment to minimize danger for other (novice) agents in its vicinity. Through an approach focusing on action prediction strategies for agents with unknown goals, I create a model capable of using expert demonstrations to learn typical behaviors for a multitude of tasks. I then apply this model to predict likely agent behaviors and identify regions of risk within this action space. This information is used to prioritize and execute risk mitigating behaviors, manipulating and adapting the environment to minimize the potential harm the novice is likely to encounter. The chapter concludes with an evaluation using multiple agents of varying goal-directedness, comparing agents' self-interested performance in scenarios with and without the assistance of a caregiver incorporating the presented model. The experimental results are promising, demonstrating that assisted agents incur less damage, interact longer, and explore their environments more completely than unassisted agents.

3.5.1 Motivation

Caregiver-guided and free exploration activities are known to be an important part of development. It has been shown that the interpretation of goal-directed spatial behavior as intentional action can begin as early as 12 months [163], suggesting that caregiver behaviors may impart high-level information to infants. The same study shows 12-month-olds to be even capable of evaluating an action's rationality in limited circumstances, indicating a preliminary understanding of goal-directed behaviors. Particularly for children between one and two years of age, exploratory activity is critical to development [164].

Earlier work in examining exploratory behaviors of infants and toddlers has shown there

to be direct correlations between these behaviors and problem-solving ability [165]. At 12 months of age, studies indicate that a greater breadth of exploratory behavior, when faced with novel objects (toys), was linked to both an increased quantity of behavior as well as more successful and sophisticated problem-solving ability. These connections continue into toddler years, as others [164] have shown that exploratory style at 19 months of age is predictive of typical vs. delayed developmental level (measured by pretend play level and performance of meaningful actions) at 30 months, painting a more complete picture of developing competence (effective adaptation).

Two-year-olds inherently include social interactions in their exploratory behaviors, learning about the world around them. For example, the presence of a toddler's mother during a novel exploration task is known to elicit social behaviors, creating active and passive bids for involvement and support of action during exploration. Children that make more bids to their mothers during exploration do so more actively than those that do not [166, 167]. This suggests that a trusted caregiver may be able to encourage deeper and more plentiful interactions.

Toddlers commonly encounter issues where goal-directed exploration and learning is limited by the presence of environmental hazards. Situations can occur in which a caregiver might wish a toddler to learn to avoid a particular action (e.g., touching a hot stove-top) without the toddler experiencing the environmental punishment for doing so (e.g., receiving a painful burn). Another common scenario may involve discouraging a child from exploring or playing near stairs or other falling hazards, without him directly experiencing the potential negative consequences of interacting in such an area. The trauma from experiencing such hazardous events can have lasting developmental effects, persisting even through adulthood [168].

In this section, I propose a novel shaping mechanism by which a robot can reduce a novel agent's inherent risk during exploration within an environment. The proposed method accomplishes this without sacrificing the quality of the learning experience, producing behavior for an assistive robot that can autonomously adapt a novice agent's environment to reduce the risk of damage to themselves or to objects nearby. To do so, this robot must monitor the behavior of the novice, predict intended goals, and take non-invasive, preventative measures

to ensure the exploring agent's safety.

I accomplish this without any direct communication, as all information is implicitly conveyed through manipulations to, and exploration within, the environment. This allows for the application of the contribution to agents of unknown capabilities and with agents incapable of direct communication. As an example, if a learner is at risk of collision with any objects, those objects should be protected accordingly (perhaps by placing a barrier around them or temporarily relocating them somewhere safer).

To this end, I extend previous work on how to autonomously give advice to an agent that may be lost, may be exploring its surroundings, or may have a suboptimal policy [169]. Drawing on these ideas, I develop and present a model allowing an assistant or caregiver observing the environment to learn a model of normal behaviors from successful trajectories through the space. Deciding if a new agent moving through the environment needs assistance is then a form of anomaly detection, by determining how well that new agent fits the behavior predicted by the model.

This assistance must take the form of changes to the environment, such as moving a fragile or hazardous object to a less risk-prone area or adding a temporary barrier to that area, to remain minimally invasive. This indirect assistance is offered in opposition to more traditional modes of direct assistance that use physical manipulation, where the assistant's action policy is imposed [121] (for instance, by being led by the hand through a task). The proposed approach includes the generation and evaluation of indirect assistance as a means of maximizing the safety of fellow agents in an environment.

Such indirect interventions are not without precedent in the robotics or machine learning literature. The benefit of interactive shaping, the practice of providing targeted feedback to manipulate portions of an agent's policy to facilitate goal achievement, is well established [170–172]. While the ability of an agent to interpret and respond to live feedback or to freely explore a task space can greatly improve the convergence rate of its action policy, there exist many cases where catastrophic failures may occur. In the chosen application domain, these failures are unacceptable to use as learning opportunities as they may result in disastrous injury.

Accordingly, a novice agent's risk is modeled with the expectation of perturbation from

typically observed behaviors derived from expert demonstrations. The presented approach accomplishes this by probabilistically merging collections of task-based risk models, and choosing environmental manipulations in order to minimize the expected risk given the potential execution policies of the agent.

The lack of direct communication between novice and caregiver introduces the requirement of estimating the probability that an agent is behaving safely based on its trajectory. The assistant may then adapt the environment to mitigate the risks inherent to the nearest danger zones. I demonstrate the utility of the presented approach by evaluating its effectiveness at providing a safe interaction environment for agents with uncertain or suboptimal navigation policies and behaviors.

In the following section I introduce term definitions and necessary background information. I then describe the process of building models to assess safety and risk given collections of expert demonstrations for a variety of tasks. Finally, the section concludes with an evaluation within a simulated online risk mitigation domain, validating the proposed approach through various metrics related to safe exploration and goal achievement for agents of varying levels of goal-directedness.

3.5.2 Preliminaries

Throughout the remainder of this section, I employ the following terminology. The *novice* is moving around the environment, the *caregiver* watches the behavior of the novice and adapts the environment, and *objects* are regions with which the novice can collide. Such a collision causes *damage*, which can be treated as a negative reward in the context of reinforcement learning. To prevent damage, the caregiver manipulates the environment such that the novice still negatively reinforces undesirable behaviors but without incurring the full damage that such a behavior may have otherwise inflicted.

Let an environment be specified by a set of states S, and the actions available to an agent are drawn from a set A. An agent in a state $s \in S$ can select an action $a \in A$, after which the agent transitions to another state $s' \in S$ according to a probability distribution given by the transition function $T(s, a, s')$. This transition results in a reward given by the reward function $R(s, a)$. The motion of an agent through the environment is thus given by

the Markov decision process (MDP) (S, A, T, R) [173].

A decision rule for probabilistically (or deterministically) selecting actions as a function of state is a policy $\pi : S \times A \longrightarrow [0, 1]$. An optimal policy π^* is the policy which maximizes total reward from every state of the MDP.

Action priors [174] are a recently proposed mechanism for learning models of behavior in a common domain across multiple tasks. These involve maintaining distributions over the action set for each state, corresponding to the number of known optimal policies that select each action at that state. Particularly for domains in which goals cannot be or are not fully specified, I include in the set of optimal policies the set of 'best known' policies for a given task. This provides a basis for learning goals from inverse reinforcement learning [2] approaches, where expert demonstrations can train a reward function in lieu of a specified goal state descriptor.

As such, for each state s in the environment, the action prior $\theta_s(a)$ is computed from $\alpha_s(a)$, the number of optimal policies π in the set of all such policies Π in which a is taken in s with probability greater than some threshold δ. Thus,

$$\alpha_s(a) \quad = \quad \|\{\pi \in \Pi | \pi(s, a) > \delta\}\| + \alpha_s^0(a),$$

where $\|\{\cdot\}\|$ represents the size of a set, and $\alpha_s^0(a)$ is a hyperprior used to prevent overfitting [174], by allowing for that fact that the training policy set Π may not be fully representative of the set of all behaviors in the environment.

The action priors are then typically computed as a draw from a Dirichlet distribution $\theta_s(A) \sim \text{Dir}(\alpha_s(A))$. This state-based distribution over the action set assigns probability to each action based on the number of tasks for which that state-action combination was optimal. This therefore provides a model of optimal behavior in the environment, aggregated across all tasks.

3.5.3 A Model of Safety

I approach the problem of facilitating safe learning and exploration initially as that of anomaly detection. Given a model of safe interaction, described by the set of MDP reward

functions within known tasks and encapsulated as action priors, observed agent behaviors are compared to these known policies. This model construction is particularly robust as it makes determinations independent of an agent's goals, perceptual capabilities, or assumed knowledge of the world. By using such an approach, the pitfall of relying on constructing models of unsafe interaction is avoided, which may not be feasible or reasonable to construct.

Computing the probability that a trajectory was drawn from a model of abnormal or risky behavior is desirable, but to learn such a model one would need examples of this (which is undesirable from the point of view of both the agent and the environment). Instead, the probability of the trajectory having been drawn from a model of normal behavior is used as a proxy, $P(trajectory|model)$, which can safely be learned from experience.

This model requires computing the probability that a novice agent is behaving safely. Let the trajectory taken by the novice thus far be represented by an alternating state and action sequence, as

$$\tau \quad = \quad s^{t+1}, a^t, s^t, a^{t-1}, s^{t-1}, \ldots.$$

The probability that a novice is behaving safely is computed as the probability that the trajectory followed by the novice was drawn from a model of normal motion in the environment, as modelled by the action priors. By Bayes' rule, this is given by

$$P(safe|\tau) \quad = \quad \frac{P(\tau|safe)P(safe)}{P(\tau)} \tag{3.2}$$

$$= \frac{\prod_{k=1}^{t} \theta_{s^k}(a^k) P(safe)}{\prod_{k=1}^{t} \theta_{s^k}(a^k) P(safe) + \prod_{k=1}^{t} \rho_{s^k}(a^k)(1 - P(safe))}$$

where $P(safe)$ is the prior belief that the agent is choosing a safe action at each timestep, and $\rho_{s^k}(a^k)$ is an action distribution for an unsafe agent.

As the model of unsafe behavior is not known *a priori* and it is typically infeasible to collect this data, I represent this model as one where every action is assumed to occur with equal probability. A richer model of goal likelihoods could improve upon this, however doing

so imposes requirements of deep domain knowledge and rich observation on the caregiver to collect or use such data. As such, in the experiments $\rho_s(a) = 1/|A|$, $\forall s \in S, a \in A$. Furthermore, without *a priori* knowledge, a uniform prior is assumed over the probability of the agent behaving safely with $P(safe) = 0.5$.

3.5.4 A Model of Danger

Having computed a probabilistic estimate of how unsafe the behavior of a novice agent is given a model of expert motion, the caregiver is required to modify the environment to enhance the safety of the agent.

Damage is caused by a collision between an agent and any of a number of objects in the environment. The caregiver must therefore estimate which object poses the greatest risk to the novice. To do so, it computes the expected damage caused by a collision with each object o. This is determined as the damage that would be caused by a collision between the novice and that object, weighted by the probability of that collision, as

$$\begin{aligned} \mathrm{E}(d_o|\tau) &= P(collision|\tau) \times d_o \\ &= (1 - P(safe|\tau)) \times P(reach_o|\tau) \times d_o \end{aligned}$$

where $\mathrm{E}(d_o|\tau)$ is the expected damage that could be caused by a collision with object o given the current trajectory τ of the novice, $P(safe|\tau)$ is the probability of the novice behaving safely as given by Equation (3.2), $P(reach_o|\tau)$ is the probability the novice reaches and collides with o, and d_o is the extrinsic cost of the damage caused by such a collision. As a proxy for $P(reach_o|\tau)$, this quantity is represented by the normalized distance of the agent from o. This provides an estimate of the number of timesteps that would be required for the novice agent to reach and collide with the object from its current position.

The damage cost d_o is extrinsically defined, and associated with different items in the environment based on the danger that a collision with an agent may pose to either the agent or the object. An expected intrinsic cost to the agent must be computed, based on the distance of the agent from that object, as well as the estimated probability of a collision.

The caregiver must follow an action policy balancing these costs, subtracting the cost of

a particular danger mitigation strategy from the potential harm. The cost to mitigate risk is considered proportional to the time required for the caregiver to perform the environmental modification.

3.5.5 Experiments

I validate this approach in an experiment simulating a novice agent exploring a household domain. In each goal-directed episode the novice agent is attempting to navigate to a sequence of randomly selected toy bins within the environment. In addition to walls and toy bins, this environment contains several hazards, some mobile and others immobile. Hazards which may not be moved include stairwells and tables, while those that can be relocated are candles that sit upon tables. I explore the behavior of a caregiver robot by examining its interactions with different novice agent types, varying the novice's goal-directedness via exploration likelihood, as they interact with the environment and perform various navigational tasks (in the form of object retrievals).

3.5.5.1 Environment

The experiment utilizes an environment characterized by a house's floor plan, discretized into a 2D grid world. Apart from walls and free space, the environment has five types of objects: candles on tables which may start fires if collided with (major damage), walls and tables which may be bumped into (minor damage), stairs that the agent may fall down (major damage), toy bins that act as goal locations for the novice, and the mobile caregiver robot. The domestic environment used in the experiments is shown in Figure 3.13.

3.5.5.2 Novice Agent Behavior

Novice agents are initialized at a random start location with knowledge of all toy bin (goal) locations, though the rest of the environment is unknown. Without a goal, a novice will randomly choose a goal destination from the list of known toy bins. When a goal has been selected, the novice agent will follow an $\epsilon-$greedy policy to achieve it before selecting a new goal, meaning it will take an optimal action $\epsilon\%$ of the time and a random action $(1-\epsilon)\%$ of the time. In the simulation, I allow novices to 'play' for a maximum of 200

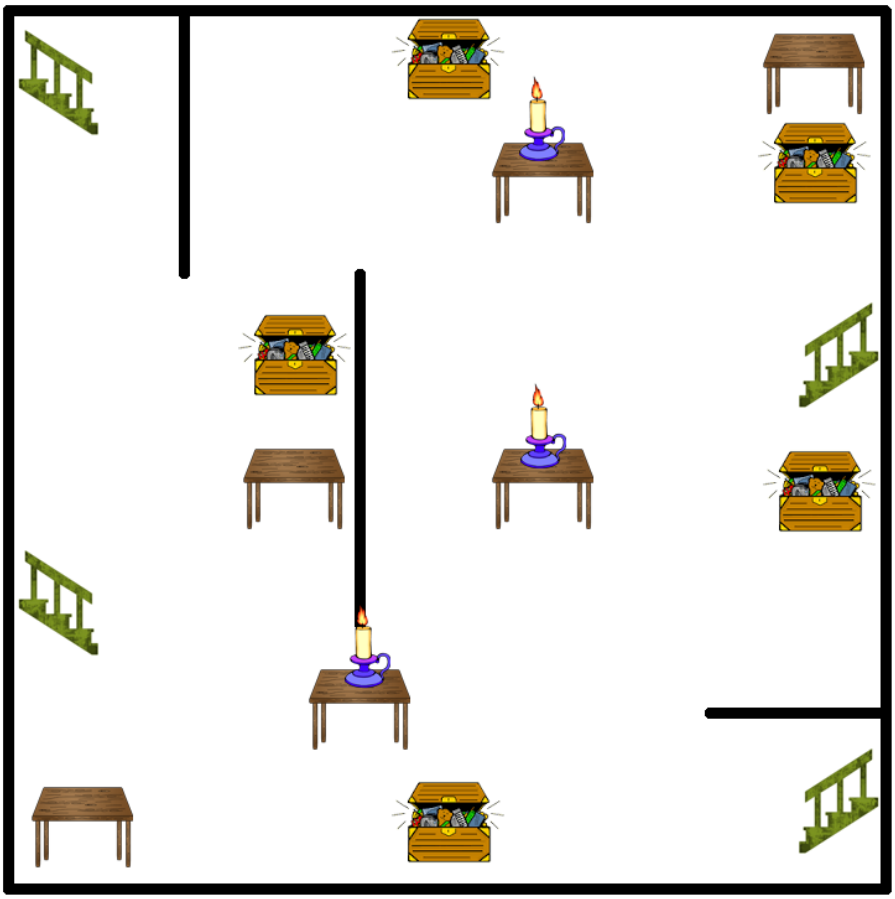

Figure 3.13: A map used in the experiments. The environment consists of open spaces, walls, staircases, tables, candles, and toy bins.

timesteps, with each timestep corresponding to the time required to move one grid square in the environment. The action set for these agents is limited to 4-connected movement within the environment. Collisions with objects or walls result in the agent staying in place and incurring a penalty. The amount of damage is extrinsically defined as being 5 for a collision with a stairwell, 4 for a collision with a candle, and 1 for a collision with an empty table. Collisions with walls or toy bins do not incur any damage.

3.5.5.3 Caregiver Training and Behavior

The model of normalcy I utilize is developed from a number of expert trajectories of an optimal agent moving through the room. The expert trajectories take the form of collision-free, shortest-path routes from various start locations to each of the specified goal locations.

These optimal policies provide the action priors required to inform the risk mitigation strategy of the caregiver. The evaluation simulates a caregiver robot that executes up to 3 actions for each 1 of the agent, assuming a movement speed ratio similar to that of healthy adult humans (5.0 km/h) [175] to healthy 1-2 year olds (1.6km/h) [176]. The action set for the caregiver consisted of 7 actions: 4-connected grid movement operations, a wait action, a get-object action, and a place-object action.

There are two intervention strategies which can be employed by the caregiver. Firstly, it is able to pick a candle up off a table, and move it to another. Secondly, the caregiver may wait in a stairwell, which blocks the novice from accessing it and subsequently incurring a large amount of damage.

3.5.6 Evaluation Criteria

The performance of the supportive behavior algorithm is characterized by the average damage incurred by a novice agent over the course of its interactions (Figure 3.14 and Table 3.1), the number of timesteps elapsed before the novice reached a critical damage threshold that terminates the simulation episode (Figure 3.15 and Table 3.2), and the amount of environmental coverage the agent was able to achieve during exploration (Figure 3.16 and Table 3.3). As the goal of this algorithm is to actively mitigate and minimize the risks inherent to interaction and exploration in a novel environment, the average incurred harm is an essential metric to track. Examining how much time the novice is able to use for exploration/task execution in the environment, before it is forced to stop as a result of accumulating too much damage, is also crucial for characterizing this contribution. The final and most relevant metric I use for analysis is the amount of environment coverage that the novice achieved (equivalent to environmental knowledge gained). This measurement shows how well the supportive behaviors allowed the novice agent to better inform its navigation and other relevant interactions in the task.

The results reported in this section are all averaged over 20 runs (episodes), each of the novice agent interacting in the environment both with and without the caregiver robot present. As the values I present will change with different environments, objects, and damage definitions, the trends of the results are more important than the particular values

themselves.

3.5.7 Results

These results show definitive and tangible benefits achieved by the risk mitigation algorithm described within this section. Novice agents with caregivers present received less damage from the environment (Figure 3.14), spent more time interacting (Figure 3.15), and explored more of the environment before receiving a critical amount of damage (Figure 3.16) than those without caregivers.

In Figure 3.14, I plot the average damage incurred by the novice over each of its interaction episodes, lasting 200 timesteps. It is expected that an agent with higher exploration rates will encounter more harmful areas of the environment, and as such experience more damage. The inclusion of the caregiver agent generally reduces this incurred damage by over 66%, performing mitigation strategies informed by known optimal behaviors and predicted deviations.

While each episode lasted 200 timesteps, it was also recorded when an agent sustained above a critical threshold of damage (corresponding to 15, or 3 times the damage of a staircase collision). I examine the number of timesteps the agent is able to complete, before either sustaining too much damage to continue or reaching the end of the episode (fixed at 200 steps) in Figure 3.15. Understandably, less goal-directed novice agents are more prone to encountering harmful objects in the environment. On average, fully random and 75%-random agents without caregiver assistance do not even complete a quarter of the episode before sustaining critical damage. Unassisted agents with ϵ-greedy strategies of 50% and 25% often only complete half of the episode. Even with a 5% exploration rate, the agent in the non-caregiver condition does not always complete the episode before sustaining critical damage, illustrating the danger inherent within the test scenario environment.

When introducing and training the proposed caregiver agent, the number of timesteps before sustaining critical damage improves substantially. The caregiver mitigates risk even in the fully random novice, an agent entirely without goal directed motion, by doubling the number of actions taken prior to episode termination. Goal directed novices experience even more dramatic improvements, with 75%-random and 50%-random agents completing

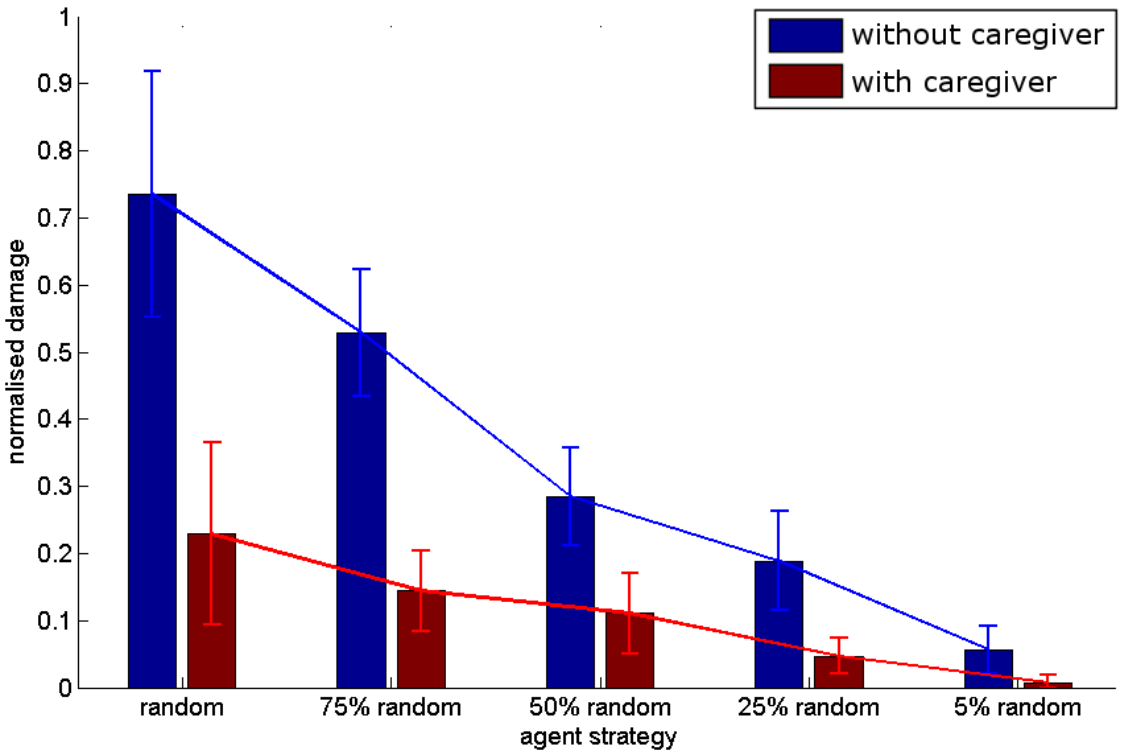

Figure 3.14: The average normalized damage (negative reward) accumulated by novice agents of various levels of goal-directedness, both with and without the assistance of a caregiver robot. Error bars represent one standard deviation from the mean. A value of 1.0 on this graph corresponds to the maximum damage received in a single episode across all agent types and conditions. Novices followed an ϵ-greedy policy to their goal, determining the frequency of choosing optimal or random exploratory actions.

over 150 of the 200 possible timesteps on average, with many of them completing the entire episode without receiving critical damage. For 25%-random and 5%-random agents the caregiver robot was able to ensure *every episode* ran to completion, with no agents having their interaction terminated early. These results are indicative of the clear benefits afforded by the caregiver's control policy, achieved through accurate safety prioritization and action prediction.

Finally, I examined the overall environment coverage achieved by the various novices prior to reaching the damage threshold (Figure 3.16). It is fairly straightforward to expect that a weakly goal-directed agent (one with a high exploration rate) that is able to spend more time interacting in the environment will likely cover more ground than a strongly goal directed or temporally limited agent. Accordingly, agents attain task space coverage that

Agent	No caregiver	With caregiver
random motion	0.7346	0.2291
75% random	0.5295	0.1438
50% random	0.2846	0.1106
25% random	0.1887	0.0466
5% random	0.0565	0.0072

Table 3.1: Mean normalized damage (negative reward) accrued by various novice agents with and without caregiver intervention

Agent	No caregiver	With caregiver
random motion	35.5000	97.0500
75% random	41.0500	164.2000
50% random	89.5000	171.5000
25% random	107.7000	200.0000
5% random	195.7500	200.0000

Table 3.2: Average number of timesteps until threshold damage with and without caregiver intervention

varies in proportion to both the exploration rate and episode duration.

In the non-caregiver condition, the entirely random novice agent covers fewer than 15 cells on average, with the 75%-random agent not faring much better. The 50%-random agents, using nearly double the actions on average to explore before sustaining critical damage, achieve approximately double the environment coverage as their less directed counterparts. Finally, the 25%-random and 5%-random agents perform best, reaching nearly 35 states on average.

When interacting in an environment with the caregiver robot, the novices generally outperform their solo counterparts with only the 5%-random agent having similar results (largely due to its limited deviation from an optimal trajectory). This coverage increase is directly attributable to the increased duration of exploration available to the agent. I include these results to provide a concrete measure of the realized benefit from having the additional time for environmental exploration and interaction. I show that not only does the caregiver robot provide a safer environment, but this risk mitigation results in actual increases in environmental exploration, and subsequently potential increases in the agent's experience diversity and its awareness of its surroundings.

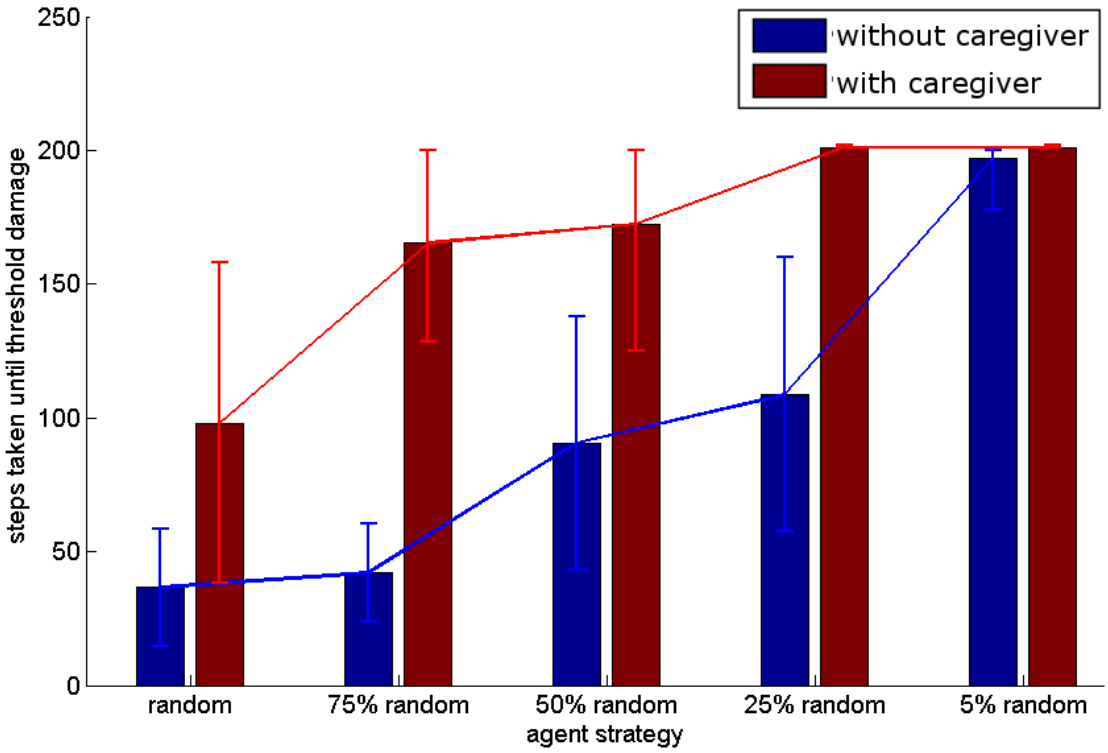

Figure 3.15: The average number of timesteps (maximum 200) before the novice agent incurred damage above the stoppage threshold. Error bars represent one standard deviation from the mean. Values of 200 indicate that the agent completed the entire interaction without incurring damage above the termination threshold.

3.5.8 Discussion and Summary

I present a model usable by a caregiver agent to assist novice agents through the reduction of danger inherent in an environment. This is accomplished through the utilization of expert demonstrations to learn regular behaviors for multiple tasks within an environment. From this data I create a model of normal behaviors, using it to enable an autonomous agent to mitigate the highest-risk scenarios the agent it's assisting is predicted to encounter. In doing so I show how an environment can be adapted online to respond to the assisted agent's behaviors and goals in the absence of directly communicated information specific to the agent's intent.

The presented results show that this approach is effective in reducing harm experienced by a novice agent interacting in the same environment as the caregiver. This manifested

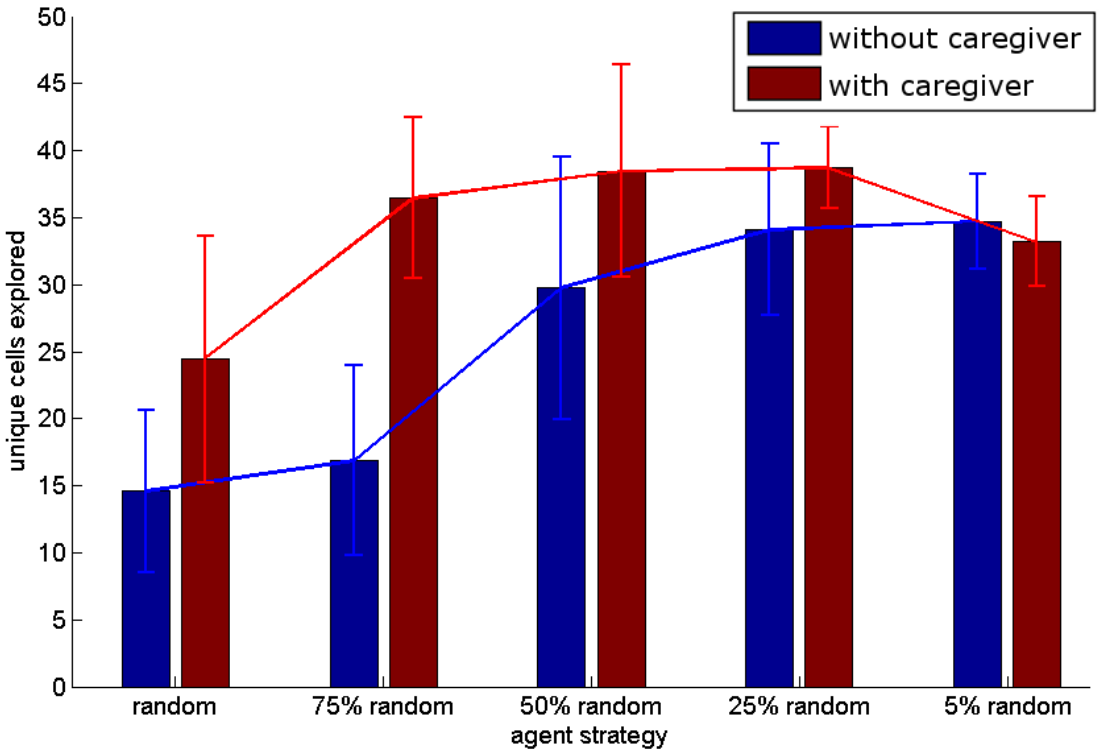

Figure 3.16: The average number of unique grid cells the novice visited per episode. Error bars represent one standard deviation from the mean. The number of cells visited is used as a metric to evaluate the amount of environmental knowledge gained by the novice agent over the course of an episode (up to 200 timesteps).

other important benefits, including increased interaction and exploration time (prior to receiving a critical amount of damage) as well as increased coverage of the environment. The trends within the data demonstrate the value of such a caregiver, particularly due to its lack of reliance on direct communication or knowledge about a specific agent.

Providing a means of autonomously adapting an environment for a novice agent to make exploration less risk-prone is valuable within many contexts. In addition to mitigating risk for novice humans, the same approach can be used to assist robots learning tasks within an environment. The same benefits persist, as a robot that is able to perform more iterations of an action or explore the action space more completely without damaging itself would be expected to achieve better task proficiency than one without these advantages.

Agent	**No caregiver**	**With caregiver**
random motion	14.6000	24.4500
75% random	16.8500	36.4500
50% random	29.7500	38.4500
25% random	34.1000	38.7000
5% random	34.7000	33.2000

Table 3.3: Average number of cells explored with and without caregiver intervention

Chapter 4

Conclusion and Future Directions

In this thesis I have presented methods by which a robot collaborator can learn rich, hierarchical task representations, then act upon them within multi-agent contexts to improve the safety, efficiency, and skill proficiency of its teammates.

In the previous chapters I have presented robot controller algorithms that support human workers in shared-environment collaborative tasks. I have developed algorithms that autonomously build rich, hierarchical representations of tasks from human demonstrations that, unlike traditional sequential task models, explicitly encode structural information including which subtasks can be performed in parallel, which subtasks must be completed sequentially, and which agents can perform a particular subtask. Using these representations, I developed algorithms providing a collaborative robot with the capability to anticipate the needs of a human teammate and proactively offer help via supportive behaviors. This included demonstrations of robots that provided tools just in time, that altered the workspace to make certain task orderings more feasible, and that recognized when a user was delayed in a complex task and offered assistance.

4.1 Summary of Contributions

The individual contributions of the thesis are summarized below:

- A novel method of autonomously constructing hierarchical goal networks from demonstration and exploration

- An active learning-based exploration strategy informed by task-graph topology that achieves accelerated task comprehension
- An approach to policy transfer between humans and robots for learning supportive behaviors from natural language
- A task and motion planning approach to discovering and executing supportive behaviors in arbitrary tasks
- A method for achieving real-time intention recognition and role inference utilizing HTNs and low level motion cues
- A robust, goal-free approach to autonomously facilitating others' self-directed learning while maintaining their safety, using existing domain expertise

4.2 Future Work

Although the task representation I developed in this thesis allowed the robot to interpret activity of the human collaborator, the robot's ability to be a successful collaborator was limited by its inability to respond to the humans attempts to request assistance and to ask the human worker for clarification on tasks in progress beyond a superficial level. The logical extension of work presented in this thesis is to develop robots that are able to respond to natural language requests for role allocation or assistance, to query the human partner to resolve ambiguity, and to respond proactively to clarify its own actions when (and only when) it judges that they would appear to be ambiguous or unclear to the human user.

4.2.1 Resolving uncertainty in collaborative tasks

While many current robotic and cognitive architectures allow for uncertainty to help guide action selection, most of these systems treat uncertainty passively the uncertainty of the world is to be modeled and used to select appropriate actions from an existing repertoire. A more informed approach would be to both represent uncertainty explicitly and, when possible, to resolve uncertainty by taking high-level actions (e.g., by communicating with other agents as opposed to performing motor actions and exploring the task space directly).

Without interrupting progress toward the mutual goal or introducing excessive cognitive load, such a system could use dialogue to resolve ambiguities about human collaborators' intentions and current activities, (re)confirming the situational model of the task as it evolves, communicating events and states which may be perceptible only by the human or only by the robot, and keeping the human partner informed about the robots actions and (unobservable) decisions.

Most grounded dialogue systems in use in HRI research currently operate exclusively on user-driven commands to the robot. While this format is appropriate for a master/slave relationship, it relies on a set of assumptions that must be broken when dealing with collaborative human-robot teams. To engage in communication with its human partner, a collaborative robot must be capable of processing short fragments that are requests for actions (commands), requests for information (inquiries), informative statements, and even off topic utterances. These fragments can often only be understood in context we ask our partners to hold this still, give it here, or simply just mutter over there or a bit to the left. To engage in this dialogue, language representations must be grounded not only in the physical world but also within the representations of the collaborative activity itself. A key benefit of viewing communication within the context of collaborative task execution is that this dialogue can help to simplify some of the many tasks that the robot must perform by reducing uncertainty.

4.2.2 Task Monitoring and State Estimation

In order to be successful, collaborative robots must combine a set of representation systems that have been developed in different research communities. Most robotic architectures treat task models as sequential activities that can be modeled through probabilistic state-transition models (such as MDP variants). These architectures can learn task models from observed action sequences (learning from demonstration), but typically offer only opaque abstractions which cannot be easily communicated and are not suitable for a broader repertoire of cognitive operations. In contrast, the knowledge substrate of a language endowed agent must include descriptions of a wide variety of objects, events, and relations which can be connected to ways of expressing them in a natural language. As a result, ontologies

and lexicons supporting such systems cannot within the current state of the art be predominantly induced from experience at necessary scale. Unlike most dialogue systems, a real-world robotic implementation must address the very complex problems of reference resolution (textual co-reference and, even more importantly, grounding references to elements of agent memory).

One of the primary tasks that a collaborative robot must perform is to continuously monitor which sub-task its partner is performing, and often assess how far along the partner is toward the completion of that sub-task. Using hierarchical task models, this corresponds to recognizing which leaf node of the task tree the user is currently engaged. By monitoring the progress of the human partner, a robot can make intelligent choices about which subtasks it should engage in (to prevent conflicts of resources or duplication of effort), can offer assistance if a task is taking abnormally long, and can collect usage data on the preferences and abilities of its partner. Current estimation mechanisms are based off intention and activity recognition which typically use motion-based metrics or statistical classification techniques. A natural extension of these techniques includes allowing the robot to ask questions of the human partner in order to reduce uncertainty over the robot's state-belief distribution.

As an example, suppose that a human-robot team is assembling a flat pack chair. Many of the components are nearly identical (such as the left and right supports), differing only in the placement of one or two holes along the length of the piece. If the human user begins assembling the left support, the robot can either assist the user with that task or begin work on a parallel task (such as the right support). If the user is relatively skilled, performing a secondary task in parallel is a more efficient use of the team resources. However, in order to begin working on a parallel task, the robot must be certain which side the user is working on. Within the hierarchical model framework presented earlier in this thesis, much of the uncertainty of perception and activity recognition comes from the inherent similarities of atomic tasks. Grasping a screw from a storage bin or retrieving a tool may look exactly the same regardless of which subtask is being executed. While some ambiguity can be resolved based on the robots knowledge of user preferences and the historical activity context, this uncertainty in user state estimation is likely to remain high. The use of interactive human-

robot dialogue can be leveraged to achieve high resolution, low-latency state estimation.

4.2.3 Converging upon shared mental models

A great deal of work so far has assumed that human and robot coworkers have identical and accurate conceptions of the task to be performed. This assumption is clearly not viable for many real-world situations. Humans and robots will vary in their understanding of their task, and such discrepancies can result in a variety of failures. A truly collaborative system must have the ability to continuously adapt to differences between its concept of the task and its partners task understanding.

Efficiently and safely identifying misalignments between a robot's hierarchical task representation and a human teammate's conceptualization of the task is a challenge that remains an open problem. Beyond merely identifying these disconnects in understanding, the process of resolving them is arguably an even more difficult problem, possibly requiring the robot to make inferences about the meanings of previously unknown actions, words, and phrases used by a partner.

Once a misalignment has been detected, there are at least three types of actions that might potentially be taken, based on the relative proficiencies of the human and robot on the task in question:

1. *Robot as expert:* If the robot is a task expert, it should explain itself to the human user in an attempt to correct their view of the task. The process of generating English dialogue explanations of the robot's current hierarchical task plan, providing concise explanations of how the user's actions differ from those expected by the robot's understanding of the task, is a challenging research problem with great value to the collaborative robotics research community.

2. *Robot as novice:* If the robot is a novice, it should ask questions of its coworkers that both clarify its understanding of its coworkers' mental model and help to provide a plan for modifying the robot's own task model.

3. *Robots role is unclear:* If it is unclear who is the more proficient team member, a negotiation should take place. This negotiation process is likely to require deep

domain knowledge and a great degree of expressivity to achieve, but is also a critical research challenge for human-robot collaboration that warrants attention.

Bibliography

[1] G. Konidaris, S. Kuindersma, R. A. Grupen, and A. G. Barto. Autonomous skill acquisition on a mobile manipulator. In *AAAI Conference on Artificial Intelligence*, 2011.

[2] A. Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, pages 663–670, 2000.

[3] N. T. Nguyen, D. Q. Phung, S. Venkatesh, and H. Bui. Learning and detecting activities from movement trajectories using the hierarchical hidden markov model. In *CVPR 2005*, volume 2, pages 955–960. IEEE, 2005.

[4] L. P. Kaelbling and T. Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, pages 1194–1227, 2013.

[5] S. Nikolaidis and J. Shah. 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*, pages 33–40. IEEE Press, 2013.

[6] C. Daniel, M. Viering, J. Metz, O. Kroemer, and J. Peters. Active reward learning. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[7] J. MacGlashan, M. Babes-Vroman, M. desJardins, M. Littman, S. Muresan, S. Squire, S. Tellex, D. Arumugam, and L. Yang. Grounding english commands to reward functions. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

[8] C. Amato, G. Konidaris, A. Anders, G. Cruz, J. How, and L. Kaelbling. Policy search for multi-robot coordination under uncertainty. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

[9] R. Sutton, D. Precup, S. Singh, et al. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1):181–211, 1999.

[10] V. Groom and C. Nass. Can robots be teammates? benchmarks in human-robot teams. *Interaction Studies*, 8(3):483–500, 2007.

[11] M. C. Gombolay, R. J. Wilcox, A. Diaz, F. Yu, and J. A. Shah. Towards successful coordination of human and robotic work using automated scheduling tools: An initial pilot study. In *Proc. Robotics: Science and Systems (RSS) Human-Robot Collaboration Workshop (HRC)*, 2013.

[12] M. Gombolay, R. Wilcox, and J. A. Shah. Fast scheduling of multi-robot teams with temporospatial constraints. In *Robotics: Science and Systems*, 2013.

[13] M. Gombolay, R. Gutierrez, G. Sturla, and J. Shah. Decision-making authority, team efficiency and human worker satisfaction in mixed human-robot teams. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

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

[15] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In *Springer handbook of robotics*, pages 1371–1394. Springer, 2008.

[16] J. Sweeney and R. Grupen. A model of shared grasp affordances from demonstration. In *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, pages 27–35. IEEE, 2007.

[17] S. Calinon and A. Billard. Incremental learning of gestures by imitation in a humanoid robot. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 255–262. ACM, 2007.

[18] C. Atkeson and S. Schaal. Robot learning from demonstration. In *International Conference on Machine Learning*, pages 11–73, 1997.

[19] B. Akgun, M. Cakmak, J. Yoo, and A. Thomaz. 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*, pages 391–398. ACM, 2012.

[20] J. Peters, K. Mülling, and Y. Altun. Relative entropy policy search. In *Proceedings of the 24th National Conference on Artificial Intelligence (AAAI)*, 2010.

[21] A. Doerr, N. Ratliff, J. Bohg, M. Toussaint, and S. Schaal. Direct loss minimization inverse optimal control. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

[22] B. Kim and J. Pineau. Maximum mean discrepancy imitation learning. In *Proceedings of Robotics: Science and Systems*, 2013.

[23] A. L. Thomaz and M. Cakmak. Learning about objects with human teachers. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 15–22. ACM, 2009.

[24] S. Pillai, M. Walter, and S. Teller. Learning articulated motions from visual demonstration. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[25] S. Alexandrova, M. Cakmak, K. Hsiao, and L. Takayama. Robot programming by demonstration with interactive action visualizations. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[26] A. Thomaz and C. Breazeal. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 1000. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[27] C. Breazeal and A. Thomaz. Learning from human teachers with socially guided exploration. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 3539–3544. IEEE, 2008.

[28] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[29] C. J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.

[30] G. A. Rummery and M. Niranjan. On-line q-learning using connectionist systems. 1994.

[31] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003.

[32] M. Cakmak and M. Lopes. Algorithmic and human teaching of sequential decision tasks. In *National Conference on Artificial Intelligence (AAAI12), Toronto, Canada*, 2012.

[33] M. Cakmak and A. Thomaz. Designing robot learners that ask good questions. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 17–24. ACM, 2012.

[34] M. Cakmak and L. Takayama. Teaching people how to teach robots: The effect of instructional materials and dialog design. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, pages 431–438. ACM, 2014.

[35] L. S. Homem de Mello and A. C. Sanderson. And/or graph representation of assembly plans. *Robotics and Automation, IEEE Transactions on*, 6(2):188–199, 1990.

[36] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial intelligence*, 49(1):61–95, 1991.

[37] G. Konidaris, S. Kuindersma, A. Barto, and R. Grupen. Constructing skill trees for reinforcement learning agents from demonstration trajectories. *Advances in neural information processing systems*, 23:1162–1170, 2010.

[38] P. Stone and M. Veloso. Task decomposition and dynamic role assignment for real-time strategic teamwork. *Intelligent Agents V: Agents Theories, Architectures, and Languages*, pages 293–308, 1999.

[39] G. Konidaris and A. Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. *Advances in Neural Information Processing Systems*, 22:1015–1023, 2009.

[40] S. Calinon, F. Guenter, and A. Billard. On learning, representing, and generalizing a task in a humanoid robot. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(2):286–298, 2007.

[41] R. Parasuraman, T. B. Sheridan, and C. D. Wickens. Situation awareness, mental workload, and trust in automation: Viable, empirically supported cognitive engineering constructs. *Journal of Cognitive Engineering and Decision Making*, 2(2):140–160, 2008.

[42] P. S. Tsang and M. A. Vidulich. Mental workload and situation awareness. *Handbook of Human Factors and Ergonomics, Third Edition*, pages 243–268, 2006.

[43] J. Shim and A. Thomaz. Human-like action segmentation for option learning. In *RO-MAN, 2011 IEEE*, pages 455–460. IEEE, 2011.

[44] I. Menache, S. Mannor, and N. Shimkin. Q-cutdynamic discovery of sub-goals in reinforcement learning. In *Machine Learning: ECML 2002*, pages 295–306. Springer, 2002.

[45] G. Neumann and W. Maass. Learning complex motions by sequencing simpler motion templates. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

[46] C. Daniel, G. Neumann, and J. Peters. Hierarchical relative entropy policy search. In *International Conference on Artificial Intelligence and Statistics*, pages 273–281, 2012.

[47] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman. Shop2: An htn planning system. *J. Artif. Intell. Res.(JAIR)*, 20:379–404, 2003.

[48] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3):189–208, 1972.

[49] A. Mohseni-Kabir, C. Rich, S. Chernova, C. L. Sidner, and D. Miller. Interactive hierarchical task learning from a single demonstration. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 205–212. ACM, 2015.

[50] D. Glas, S. Satake, T. Kanda, and N. Hagita. An interaction design framework for social robots. *Robotics: Science and Systems VII*, page 89, 2012.

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

[52] C. Breazeal, G. Hoffman, and A. Lockerd. Teaching and working with robots as a collaboration. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1030–1037. IEEE Computer Society, 2004.

[53] G. Briggs and M. Scheutz. Facilitating mental modeling in collaborative human-robot interaction through adverbial cues. In *Proceedings of the SIGDIAL 2011 Conference*, pages 239–247. Association for Computational Linguistics, 2011.

[54] M. R. Walter, S. Hemachandra, B. Homberg, S. Tellex, and S. Teller. Learning semantic maps from natural language descriptions. In *Proceedings of Robotics: Science and Systems*, 2013.

[55] D. K. Misra, J. Sung, K. Lee, and A. Saxena. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[56] G. Gemignani, M. Veloso, and D. Nardi. Language-based sensing descriptors for robot object grounding.

[57] V. Perera and M. M. Veloso. Handling complex commands as service robot task requests. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1177–1183, 2015.

[58] V. Raman, C. Lignos, C. Finucane, K. C. Lee, M. P. Marcus, and H. Kress-Gazit. Sorry dave, i'm afraid i can't do that: Explaining unachievable robot tasks using natural language. In *Proceedings of Robotics: Science and Systems*, 2013.

[59] A. Sauppé and B. Mutlu. Effective task training strategies for human and robot instructors. *Autonomous Robots*, pages 1–17, 2015.

[60] S. Rosenthal, M. M. Veloso, and A. K. Dey. Task behavior and interaction planning for a mobile service robot that occasionally requires help. In *The AAAI workshop on Automated Action Planning for Autonomous Mobile Robots*, 2011.

[61] S. Rosenthal, J. Biswas, and M. Veloso. An effective personal mobile robot agent through symbiotic human-robot interaction. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 915–922. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[62] S. Tellex, R. Knepper, A. Li, D. Rus, and N. Roy. Asking for help using inverse semantics. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[63] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

[64] M. Bollini, J. Barry, and D. Rus. Bakebot: Baking cookies with the pr2. In *The PR2 Workshop: Results, Challenges and Lessons Learned in Advancing Robots with a Common Platform, IROS*, 2011.

[65] J. Shah and C. Breazeal. An empirical analysis of team coordination behaviors and action planning with application to human–robot teaming. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 52(2):234–245, 2010.

[66] C. Pérez-DArpino and J. Shah. Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification. *IEEE International Conference on Robotics and Automation*, 2015.

[67] J. Mainprice and D. Berenson. Human-robot collaborative manipulation planning using early prediction of human motion. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 299–306. IEEE, 2013.

[68] A. Dragan, S. Bauman, J. Forlizzi, and S. Srinivasa. Effects of robot motion on human-robot collaboration. In *Human-Robot Interaction*, March 2015.

[69] K. Strabala, M. Lee, A. Dragan, J. Forlizzi, and S. Srinivasa. Learning the communication of intent prior to physical collaboration. *Robot and human interactive communication*, pages 968–973, 2012.

[70] K. W. Strabala, M. K. Lee, A. D. Dragan, J. L. Forlizzi, S. Srinivasa, M. Cakmak, and V. Micelli. Towards seamless human-robot handovers. *Journal of Human-Robot Interaction*, 2(1):112–132, 2013.

[71] C. Crick and B. Scassellati. Intention-based robot control in social games. In *Proceedings of the Cognitive Society Annual Meeting*, 2009.

[72] B. Gleeson, K. MacLean, A. Haddadi, E. Croft, and J. Alcazar. Gestures for industry: intuitive human-robot communication from human observation. In *Proceedings of the*

8th ACM/IEEE international conference on Human-robot interaction, pages 349–356. IEEE Press, 2013.

[73] A. Dragan and S. Srinivasa. Generating legible motion. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013.

[74] A. Dragan, K. Lee, and S. Srinivasa. Legibility and predictability of robot motion. In *Human-Robot Interaction*, March 2013.

[75] V. V. Unhelkar, H. C. Siu, and J. A. Shah. Comparative performance of human and mobile robotic assistants in collaborative fetch-and-deliver tasks. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, pages 82–89. ACM, 2014.

[76] G. Hoffman and C. Breazeal. Effects of anticipatory action on human-robot teamwork: Efficiency, fluency, and perception of team. In *Human-Robot Interaction (HRI), 2007 2nd ACM/IEEE International Conference on*, pages 1–8. IEEE, 2007.

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

[78] J. Shah. *Fluid coordination of human-robot teams*. PhD thesis, Massachusetts Institute of Technology, 2011.

[79] L. Cobo, C. Isbell Jr, and A. Thomaz. Automatic task decomposition and state abstraction from demonstration. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 483–490. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[80] A. S. Clair and M. Mataric. How robot verbal feedback can improve team performance in human-robot task collaborations. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 213–220. ACM, 2015.

[81] M. Oudah, V. Babushkin, T. Chenlinangjia, and J. W. Crandall. Learning to interact with a human partner. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 311–318. ACM, 2015.

[82] R. Parasuraman, T. B. Sheridan, and C. D. Wickens. A model for types and levels of human interaction with automation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 30(3):286–297, 2000.

[83] C.-M. Huang, M. Cakmak, and B. Mutlu. Adaptive coordination strategies for human-robot handovers. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

[84] M. Salem, G. Lakatos, F. Amirabdollahian, and K. Dautenhahn. Would you trust a (faulty) robot?: Effects of error, task type and personality on human-robot cooperation and trust. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 141–148. ACM, 2015.

[85] M. Desai, P. Kaniarasu, M. Medvedev, A. Steinfeld, and H. Yanco. Impact of robot failures and feedback on real-time trust. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pages 251–258. IEEE Press, 2013.

[86] A. Xu and G. Dudek. Optimo: Online probabilistic trust inference model for asymmetric human-robot collaborations. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 221–228. ACM, 2015.

[87] K. Gold and B. Scassellati. Learning acceptable windows of contingency. *Connection Science*, 18(2):217–228, 2006.

[88] J. Lee, J. Kiser, A. Bobick, and A. Thomaz. Vision-based contingency detection. In *Proceedings of the 6th international conference on Human-robot interaction*, pages 297–304. ACM, 2011.

[89] R. Wilcox, S. Nikolaidis, and J. Shah. Optimization of temporal dynamics for adaptive human-robot interaction in assembly manufacturing. In *Proc. RSS*, 2012.

[90] J. Chen and A. Zelinsky. Programming by demonstration: Coping with suboptimal teaching actions. *The International Journal of Robotics Research*, 22(5):299–319, 2003.

[91] B. Browning, L. Xu, and M. Veloso. Skill acquisition and use for a dynamically-balancing soccer robot. In *AAAI*, pages 599–604, 2004.

[92] A. G. Billard, S. Calinon, and F. Guenter. Discriminative and adaptive imitation in uni-manual and bi-manual tasks. *Robotics and Autonomous Systems*, 54(5):370–384, 2006.

[93] S. Lauria, G. Bugmann, T. Kyriacou, and E. Klein. Mobile robot programming using natural language. *Robotics and Autonomous Systems*, 38(3):171–181, 2002.

[94] R. Toris and S. Chernova. Learning of multi-hypothesized task templates from a corpus of noisy human demonstrations. In *Proceedings of the Human-Robot Collaboration for Industrial Manufacturing Workshop at Robotics: Science and Systems (RSS)*, 2014.

[95] S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto. Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, page 0278364914554471, 2014.

[96] M. N. Nicolescu and M. J. Mataric. Experience-based representation construction: learning from human and robot teachers. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 2, pages 740–745. IEEE, 2001.

[97] M. Stolle and D. Precup. Learning options in reinforcement learning. In *Abstraction, Reformulation, and Approximation*, pages 212–223. Springer, 2002.

[98] L. C. Cobo, C. L. Isbell, and A. L. Thomaz. Object focused q-learning for autonomous agents. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1061–1068, 2013.

[99] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4, 1996.

[100] K. Erol, J. Hendler, and D. S. Nau. Htn planning: Complexity and expressivity. In *AAAI*, volume 94, pages 1123–1128, 1994.

[101] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau. Htn planning for web service composition using shop2. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(4):377–396, 2004.

[102] O. Obst. Using a planner for coordination of multiagent team behavior. In *Programming Multi-Agent Systems*, pages 90–100. Springer, 2006.

[103] L.-J. Lin. Hierarchical learning of robot skills by reinforcement. In *Neural Networks, 1993., IEEE International Conference on*, pages 181–186. IEEE, 1993.

[104] K. Currie and A. Tate. O-plan: the open planning architecture. *Artificial intelligence*, 52(1):49–86, 1991.

[105] A. Tate, B. Drabble, and R. Kirby. *O-Plan2: an open architecture for command, planning and control*. Citeseer, 1992.

[106] H. Muñoz-Avila, D. W. Aha, L. Breslow, and D. Nau. Hicap: An interactive case-based planning architecture and its application to noncombatant evacuation operations. In *Proceedings of the 16th National Conference on Artificial intelligence*, pages 870–875. AAAI, 1999.

[107] B. R. Fox and K. Kempf. Opportunistic scheduling for robotic assembly. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 880–889. IEEE, 1985.

[108] R. A. Knepper, D. Ahuja, G. Lalonde, and D. Rus. Distributed assembly with and/or graphs. 2014.

[109] N. Nejati, P. Langley, and T. Konik. Learning hierarchical task networks by observation. In *Proceedings of the 23rd international conference on Machine learning*, pages 665–672. ACM, 2006.

[110] O. Ilghami, H. Munoz-Avila, D. S. Nau, and D. W. Aha. Learning approximate preconditions for methods in hierarchical plans. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 337–344. ACM, 2005.

[111] N. Mehta, S. Ray, P. Tadepalli, and T. Dietterich. Automatic discovery and transfer of maxq hierarchies. *Proceedings of the 25th International Conference on Machine Learning*, pages 648–655, 2008.

[112] Q. Yang. Formalizing planning knowledge for hierarchical planning. *Computational intelligence*, 6(1):12–24, 1990.

[113] G. Konidaris, L. Kaelbling, and T. Lozano-Perez. Constructing symbolic representations for high-level planning. In *AAAI Conference on Artificial Intelligence*, 2014.

[114] B. Hayes and B. Scassellati. Effective robot teammate behaviors for supporting sequential manipulation tasks. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015.

[115] F. Bacchus and Q. Yang. The downward refinement property. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 286–293, 1991.

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

[117] D. Eppstein and D. Strash. Listing all maximal cliques in large sparse real-world graphs. In *Experimental Algorithms*, pages 364–375. Springer, 2011.

[118] E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter. Learning the semantics of object–action relations by observation. *The International Journal of Robotics Research*, 2011.

[119] R. A. Knepper, T. Layton, J. Romanishin, and D. Rus. Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 855–862. IEEE, 2013.

[120] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. Ffrob: An efficient heuristic for task and motion planning. In *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2014.

[121] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz. Keyframe-based learning from demonstration. *International Journal of Social Robotics*, 2012.

[122] C. Breazeal, C. D. Kidd, A. L. Thomaz, G. Hoffman, and M. Berlin. Effects of nonverbal communication on efficiency and robustness in human-robot teamwork. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 708–713. IEEE, 2005.

[123] B. Hayes and B. Scassellati. Challenges in shared-environment human-robot collaboration. In *Proceedings of the 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI 2013) Workshop on Collaborative Manipulation*, 2013.

[124] B. Kim, C. M. Chacha, and J. Shah. Inferring robot task plans from human team meetings: A generative modeling approach with logic-based prior. 2013.

[125] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: Extracting reusable task knowledge from visual observation of human performance. *Robotics and Automation, IEEE Transactions on*, 10(6):799–822, 1994.

[126] G. Konidaris, L. Kaelbling, and T. Lozano-Perez. Symbol acquisition for task-level planning. In *The AAAI 2013 Workshop on Learning Rich Representations from Low-Level Sensors*, 2013.

[127] M. Cakmak, N. DePalma, R. I. Arriaga, and A. L. Thomaz. Exploiting social partners in robot learning. *Autonomous Robots*, 29(3-4):309–329, 2010.

[128] C. Chao, M. Cakmak, and A. L. Thomaz. Transparent active learning for robots. In *Human-Robot Interaction (HRI), 2010 5th ACM/IEEE International Conference on*, pages 317–324. IEEE, 2010.

[129] M. Cakmak, C. Chao, and A. L. Thomaz. Designing interactions for robot active learners. *IEEE Transactions on Autonomous Mental Development*, 2(2):108–118, 2010.

[130] S. Cambon, R. Alami, and F. Gravot. A hybrid approach to intricate motion, manipulation and task planning. *The International Journal of Robotics Research*, 28(1):104–126, 2009.

[131] F. Lagriffoul, D. Dimitrov, A. Saffiotti, and L. Karlsson. Constraint propagation on interval bounds for dealing with geometric backtracking. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 957–964. IEEE, 2012.

[132] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. Ffrob: An efficient heuristic for task and motion planning. In *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2014.

[133] A. Bhatia, M. R. Maly, E. Kavraki, and M. Y. Vardi. Motion planning with complex goals. *Robotics & Automation Magazine, IEEE*, 18(3):55–64, 2011.

[134] S. Chitta, E. G. Jones, M. Ciocarlie, and K. Hsiao. Perception, planning, and execution for mobile manipulation in unstructured environments. *IEEE Robotics and Automation Magazine*, 19(2):58–71, 2012.

[135] T. Lozano-Pérez and L. P. Kaelbling. A constraint-based method for solving sequential manipulation planning problems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3684–3691. IEEE, 2014.

[136] E. Plaku and G. D. Hager. Sampling-based motion and symbolic action planning with geometric and differential constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5002–5008. IEEE, 2010.

[137] R. Alami, F. F. Ingrand, and S. Qutub. A scheme for coordinating multi-robots planning activities and plans execution. In *13th European Conference on Artificial Intelligence*, pages 617–621, 1998.

[138] M. Dogar, R. A. Knepper, A. Spielberg, C. Choi, H. I. Christensen, and D. Rus. Towards coordinated precision assembly with robot teams. *Proceedings of the 2014 International Symposium on Experimental Robotics*, 2014.

[139] A. Komenda, P. Novák, and M. Pechoucek. How to repair multi-agent plans: Experimental approach. In *Proceedings of Distributed and Multi-agent PlanningWorkshop of the 23rd International Conference on Automated Planning and Scheduling (ICAPS)*, pages 66–74, 2013.

[140] J. Hoffmann and B. Nebel. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[141] B. J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

[142] C. I. Guinn. Mechanisms for mixed-initiative human-computer collaborative discourse. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 278–285. Association for Computational Linguistics, 1996.

[143] C. Rich and C. L. Sidner. Collagen: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction*, 8(3-4):315–350, 1998.

[144] J. Rickel, N. Lesh, C. Rich, C. L. Sidner, and A. Gertner. Collaborative discourse theory as a foundation for tutorial dialogue. In *Intelligent Tutoring Systems*, pages 542–551. Springer, 2002.

[145] R. R. Murphy. Human-robot interaction in rescue robotics. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(2):138–153, 2004.

[146] K. P. Hawkins, S. Bansal, N. Vo, and A. F. Bobick. Modeling structured activity to support human-robot collaboration in the presence of task and sensor uncertainty.

In *Intelligent Robots and Systems (IROS), Workshop on Cognitive Robotics Systems*, 2013.

[147] E. Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, pages 1065–1076, 1962.

[148] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*, 2011.

[149] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[150] R. K. Brayton. *Logic minimization algorithms for VLSI synthesis*, volume 2. Springer Science & Business Media, 1984.

[151] G. Hoffman. Evaluating fluency in human-robot collaboration. In *International Conference on Human-Robot Interaction (HRI), Workshop on Human Robot Collaboration*, 2013.

[152] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, pages 79–86, 1951.

[153] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *Sixth International Conference on Computer Vision*, pages 59–66. IEEE, 1998.

[154] P. Rochat, T. Striano, R. Morgan, et al. Who is doing what to whom? young infants' developing sense of social causality in animated displays. *PERCEPTION-LONDON-*, 33(3):355–370, 2004.

[155] S. Calinon and A. Billard. Statistical learning by imitation of competing constraints in joint space and task space. *Advanced Robotics*, 23(15):2059–2076, 2009.

[156] M. Hersch, F. Guenter, S. Calinon, and A. Billard. Dynamical system modulation for robot learning via kinesthetic demonstrations. *Robotics, IEEE Transactions on*, 24(6):1463–1467, 2008.

[157] S. Bitzer, M. Howard, and S. Vijayakumar. Using dimensionality reduction to exploit constraints in reinforcement learning. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3219–3225. IEEE, 2010.

[158] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31(3):360–375, 2012.

[159] C. Crick, M. Doniec, and B. Scassellati. Who is it? inferring role and intent from agent motion. In *Development and Learning, 2007. ICDL 2007. IEEE 6th International Conference on*, pages 134–139. IEEE, 2007.

[160] L. Talmy. Force dynamics in language and cognition. *Cognitive science*, 12(1):49–100, 1988.

[161] P. Wolff. Representing causation. *Journal of experimental psychology: General*, 136(1):82, 2007.

[162] C. Crick and B. Scassellati. Inferring narrative and intention from playground games. In *Development and Learning, 2008. ICDL 2008. 7th IEEE International Conference on*, pages 13–18. IEEE, 2008.

[163] G. Gergely, Z. Nádasdy, G. Csibra, and S. Biro. Taking the intentional stance at 12 months of age. *Cognition*, 56(2):165–193, 1995.

[164] A. S. Rusher, D. R. Cross, and A. M. Ware. Infant and toddler play: Assessment of exploratory style and development level. *Early Childhood Research Quarterly*, 10(3):297–315, 1995.

[165] D. A. Caruso. Dimensions of quality in infants' exploratory behavior: Relationships to problem-solving ability. *Infant Behavior and Development*, 16(4):441–454, 1993.

[166] L. C. Mayes, A. S. Carter, and D. Stubbe. Individual differences in exploratory behavior in the second year of life. *Infant Behavior and Development*, 16(3):269–284, 1993.

[167] K. Pridham, P. Becker, and R. Brown. Effects of infant and caregiving conditions on an infants focused exploration of toys. *Journal of Advanced Nursing*, 31(6):1439–1448, 2000.

[168] B. D. Perry. Neurobiological sequelae of childhood trauma: Ptsd in children. pages 253–276, 1994.

[169] B. S. Rosman and S. Ramamoorthy. Giving advice to agents with hidden goals. In *IEEE International Conference on Robotics and Automation*, pages 1959–1964. IEEE, 2014.

[170] T. Erez and W. D. Smart. What does Shaping Mean for Computational Reinforcement Learning? *International Conference on Development and Learning*, pages 215–219, 2008.

[171] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum Learning. *International Conference on Machine Learning*, 2009.

[172] W. B. Knox and P. Stone. Interactively Shaping Agents via Human Reinforcement: The TAMER Framework. *International Conference on Knowledge Capture*, September 2009.

[173] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[174] B. S. Rosman and S. Ramamoorthy. What good are actions? Accelerating learning using learned action priors. *International Conference on Development and Learning and Epigenetic Robotics*, November 2012.

[175] R. C. Browning, E. A. Baker, J. A. Herron, and R. Kram. Effects of obesity and sex on the energetic cost and preferred speed of walking. *Journal of Applied Physiology*, 100(2):390–398, 2006.

[176] G. Cavagna, P. Franzetti, and T. Fuchimoto. The mechanics of walking in children. *The Journal of Physiology*, 343(1):323–339, 1983.